E.T.S. de Ingeniería Industrial,
Informática y de Telecomunicación

SISTEMA WEB DE VISUALIZACIÓN DE VÍDEO EN STREAMING MEDIANTE EL EMPLEO DE UNA RASPBERRY PI.

Grado en Ingeniería
en Tecnologías Industriales

Trabajo Fin de Grado

AUTOR: Javier Garcés Quílez

DIRETOR: Carlos Ruiz Zamarreño

Pamplona, fecha de defensa

upna Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

Resumen

En la siguiente memoria se realizará el estudio e implementación de un **sistema web de visualización de vídeo en *streaming* mediante el empleo de una Raspberry Pi 2B**. La utilización de esta placa de programación se debe a la obtención de un sistema de bajo coste y de altas prestaciones, puesto que esta placa permite la implementación de diversos sistemas operativos, programas y periféricos, a un precio muy razonable.

Ya que se trata de un dispositivo cuyas características son desconocidas para una gran mayoría, se va a introducir desde un inicio la Raspberry, haciendo hincapié en los **modelos y periféricos** que está dispone, y en especial en los distintos **sistemas operativos** y el rango de funcionalidad que estos ofrecen.

Tras esto, se va a enfocar la memoria hacia el estudio del **sistema operativo Raspbian**, incluyendo los lenguajes de programación que este ofrece y la **búsqueda de una programación adecuada para obtener el sistema de visualización deseado**.

Abstract

In the following memory it has been done a study and implementation of a **video streaming display web system through the use of a Raspberry Pi 2B**. This programming board has been used for achieving a low cost system with high properties. The Raspberry lets the user the implementation of different operative systems, programs and peripherals.

The characteristics of this device are unfamiliar for many people, so the beginnings of the Raspberry are going to be introduced, going through its **models** and **peripherals**, and in special, through its **operative systems** and the huge range of functionality that they offer.

Behind this, the memory is going to focus in the research of the **operative system Raspbian**, includying the available programming languages and the **searching of an adequate programming for obtaining the desired visualization system.**

Tabla de contenido

Anexo 2. Octorpint

Índice de Figuras

Índice de Tablas

Índice de Programas

1. Introducción

El proyecto realizado trata de la implementación de un **sistema web de visualización**, basándonos para ello en la programación mediante una **Raspberry pi 2B**, una placa de procesamiento portátil cada vez más empleada, debido a su bajo coste y sus altas prestaciones.

1.1. Objetivo del proyecto

El objetivo de este proyecto, como ya se ha comentado anteriormente, es la implementación de un **sistema web de visualización de vídeo en *streaming*** mediante el empleo de una Raspberry pi 2B.

Para ello se ha realizado un estudio de las diversas opciones de configuración de la Raspberry pi 2B, tanto en el aspecto de los **sistemas operativos** (S.O.), como en los ámbitos de los **lenguajes de programación** y de los **programas elegidos**.

En cuanto a los **objetivos**, se han propuesto y abordado **objetivos más simples**, para finalizar con la **unión** de dichos objetivos **para obtener el resultado final esperado**. Estos objetivos simples se han ido fijando a lo largo de la realización del proyecto y son los siguientes:

- Estudio sobre la Raspberry Pi, inicio, modelos y periféricos.
- Estudio y elección de los sistemas operativos válidos para este proyecto.
- Elección final del sistema operativo a emplear.
- Estudio de los lenguajes de programación.
- Configuración y programación inicial del módulo cámara RPi.
- Obtención de un programa más complejo para la visualización de vídeo.
- Empleo de los servomotores.
- Acceso web a los programas seleccionados
- Implementación final del programa elegido.

Por tanto el proyecto se basa en la **implementación y programación de la Raspberry Pi 2B**, y se va a completar con el **estudio detallado** de los **modelos** de esta placa de procesamiento, de los diversos **sistemas operativos** válidos y de los **lenguajes de programación** principales empleados posteriormente en la realización del proyecto.

Con ello se pretenden enseñar las **pautas teóricas** necesarias para la compresión del proyecto, a la vez que se proporcionan los **conocimientos básicos para el uso y programación** de la Raspberry por parte de cualquier usuario.

Para el desarrollo final del proyecto, tras la elección y configuración del S.O., se emplearan distintos periféricos de entrada, como la **cámara**, y de salida, como los **servomotores**. El conjunto de la cámara y servomotores sirve para la realización de una **cámara con movimiento**, para conseguir un mayor ángulo de grabación.

1.2. Estructura y fases del proyecto

Estructuralmente el proyecto se compone de 2 grandes bloques en los que se desglosa tanto las fases de implementación de un sistema web de visualización en streaming, como las bases teóricas esenciales para comprender el desarrollo de los programas.

El proyecto consta de:

- Un **bloque descriptivo y de carácter teórico** sobre los orígenes de la Raspberry pi, sus modelos y sistemas operativos, y sobre los diferentes lenguajes de programación con el objetivo de introducir los conocimientos básicos correspondientes a la implementación de la rpi y de su programación.
- Un segundo **bloque** en el que se explicará el **desarrollo de las aplicaciones** y la implementación seguida para la consecución de los objetivos.

2. Raspberry Pi

La Raspberry es una **placa de procesamiento portátil**. A estos dispositivos se les conoce como SBC (Single Board Computer), puesto que se asemejan a un ordenador completo. Se basa en un SoC (System on a Chip), que **alberga en un solo chip integrado el procesador**, **la memoria RAM y la tarjeta gráfica.**

2.1. Origen

Este ordenador de placa reducida fue desarrollado por la **Fundación Raspberry Pi** en Reino Unido. El proyecto Raspberry Pi **se inició en 2006** por la **Universidad de Cambridge**, cuando Eben Upton y compañeros suyos decidieron **solucionar el problema de la falta de formación** que mostraban los estudiantes de ingeniería informática. Sin embargo no fue hasta el año **2012** cuando **se empezaron a comercializar**.

El objetivo principal de la fundación fue **facilitar la enseñanza de informática** en colegios del Reino Unido, por lo que su fin era crear un ordenador lo más **barato** posible y que llegase al **máximo número de usuarios**.

Sin embargo, su **uso** se ha **extendido** más allá de los colegios, siendo empleada para infinidad de usos y proyectos de diferentes ámbitos, todos ellos realizables mediante pequeños proyectos de hardware y empleando un lenguaje de programación sencillo.

2.2. Modelos

Los primeros modelos que salieron al mercado consistían en 2 tipos de placa: un modelo **destinado a los desarrolladores**, el **modelo A**. Y un **modelo con mejores prestaciones**, utilizado por los demás usuarios, cuya principal diferencia se basa en que el **modelo B** dispone de conexión a internet, una premisa solicitada por los usuarios e implementada en este modelo.

Tabla 1. Comparación entre modelos A y B.

	Model A	Model B
Target Price	20,00 €	28,00 €
SoC	Broadcom BCM2835 (CPU + GPU + DSP + SDRAM)	
CPU	700 Mhz ARM1176JZF-S core	
GPU	VideoCore IV, OpenGL ES 2,0, 1080p30 Full HD HP H.264	
Memory	128 MiB SDRAM	256 MiB SDRAM
USB 2.0 ports	1	2 (via integrated USB hub)
Video outputs	Composite RCA, HDMI	
Audio outputs	3.5 mm jack, HDMI	
Onboard storage	SD / MMC / SDIO card slot	
Low-level peripherals	GPIO pins, SPI, I²C, UART	
Onboard network	none	10/100 wired Ethernet (RJ45)
Real-time clock	No clock or battery	
Power ratings	500 mA (2.5 Watt)	700 mA (3.5 Watt)
Power source	5 Volt via MicroUSB or GPIO header	
Size	85.60mm x 53.98mm	
Supported OS'es	Debian GNU/Linux, Fedora, Arch Linux	

El modelo B consta de:

- **Boardcom BCM2835 SoC a 700MHz**: se trata de un procesador creado originalmente para móviles, pero lo suficientemente potente para cumplir las funciones de software requeridas por los usuarios.
- Una memoria **RAM de 256MB**.
- **2 puertos USB**
- salidas de **video HDMI** y en analógica de video y sonido
- sistema de **almacenamiento SD**
- **conector GPIO** (General Porpouse Inputs/Outputs) de pines compatible con otros modelos.
- **10/100 Ethernet RJ45**: mediante este conector se permite la conexión a internet.
- **Fuente de alimentación**: 5 voltios con ranura microUSB, compatible con la mayoría de los teléfonos móviles.
- Como **sistema operativo** puede usar una amplia variedad de distribuciones de Linux (Debian, Ubuntu, Fedora, Arch Linux) u otros sistemas como FreeBSD o RISC OS.

Tras este modelo, surgió **en 2014** el **modelo B+,** que marca el primer gran avance en cuanto al hardware. El modelo B+ **cambia la arquitectura** de la placa, añadiendo mejoras importantes. Estas **mejoras están destinadas a la conectividad** de la Raspberry, y se presentan en forma de nuevos puertos. Se añaden 2 puertos USB más, y el GPIO pasa de 26 a 40 conectores, se modifica la ubicación de los terminales de video y se cambia la ranura para la SD, siendo en este modelo para microSD.

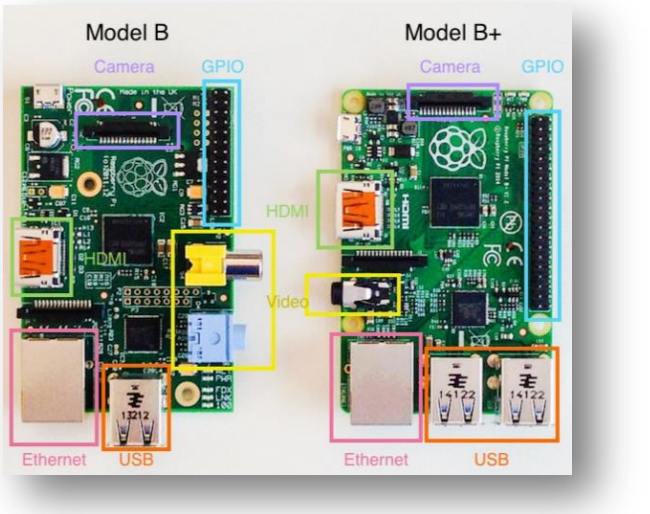


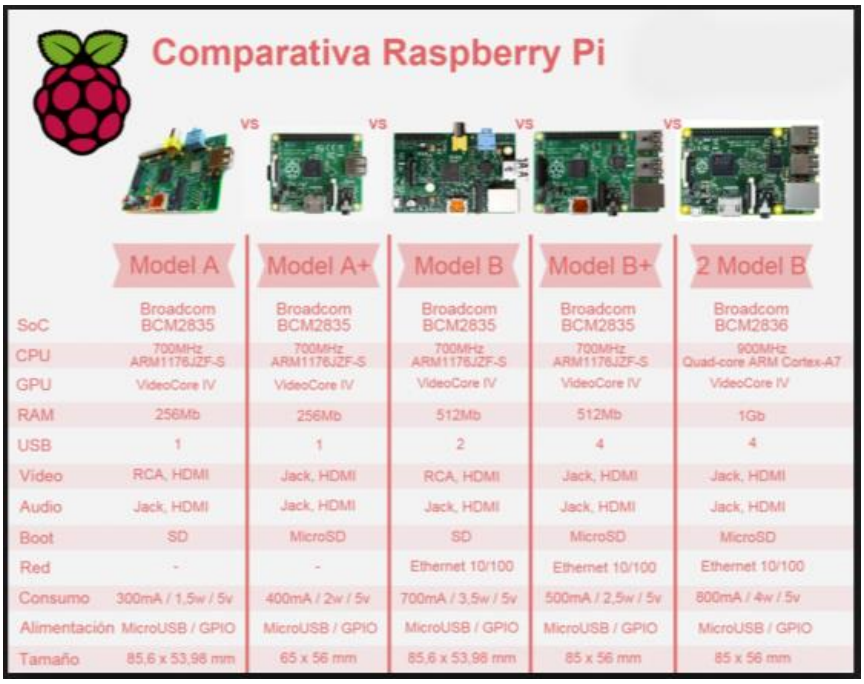
Figura 1. Comparativa entre el *hardware* de los modelos B y B+

El último modelo de esta marca, la **Raspberry pi 2**, emplea la **misma arquitectura que el modelo B+,** siendo el nuevo modelo **6 veces más rápido**. Incorpora una **mejora en la memoria RAM**, que pasa a ser de 1GB. El **procesador** cuenta **con 4 núcleos** a 900MHz con arquitectura de 32 bits, por lo que las características de la Raspberry Pi 2 incrementan considerablemente en este nuevo modelo.

Como único inconveniente de esta mejora hay que destacar el **aumento de consumo**, que pasa de 600 a 900 mA, siendo este consumo dependiente de la carga de trabajo de los 4 núcleos.

A su vez, el nuevo procesador **permite la aplicación de nuevas distribuciones y sistemas operativos,** como Ubuntu Mate o Windows 10.

Tabla 2. Comparación entre los distintos modelos de Raspberry Pi.



En febrero de 2016, ha salido a la venta el último modelo de Raspberry, el **modelo 3**. Las principales **mejoras** respecto al modelo 2B son:

- Mejora en el **procesador**, que pasa a ser: 1.2GHz 64-bit quad-core ARMv8 CPU
- Se introducen un **módulo Wifi** y un **módulo Bluetooth** 4.1.

Las demás prestaciones de esta placa son similares al modelo anterior, además de poseer el mismo hardware.

El **modelo empleado** en este proyecto es la **Raspberry Pi 2B**, que se trataba del último modelo de la marca al comienzo del mismo. Ya que la comercialización de la última edición de la Raspberry Pi empezó a comercializarse hace pocos meses, el modelo 2B sigue siendo el más empleado por los usuarios y del que más información puede encontrarse.

2.3. Sistemas operativos

Una vez mostrados los diferentes modelos y características de la Raspberry Pi, hay que realizar la **comparativa y elección del sistema operativo** que se va a emplear. Como ya se ha comentado anteriormente, el nuevo procesador de la Raspberry Pi 2 abre las puertas de diferentes sistemas operativos.

2.3.1. Definición de S.O.

Un sistema operativo puede ser definido como un **conjunto de programas** especialmente diseñados para la ejecución de varias tareas, que **sirven de intermediario entre el usuario y el procesador.**

Las **funciones básicas** de un S.O. se pueden resumir en 5:

- Inicializar el hardware del ordenador
- Suministrar rutinas básicas para controlar dispositivos
- Permitir administrar, escalonar e interactuar tareas
- Controlar los periféricos
- Mantener la integridad de sistema

El Sistema Operativo puede ser almacenado en un disco, y determinadas partes de él son cargadas en la memoria del ordenador (RAM) cuando es necesario. En el caso de la Raspberry Pi, el sistema operativo **se instala en una tarjeta microSD**.

El sistema operativo provee utilidades para:

- Administración de Archivos y Documentos creados por usuarios
- Ejecución controlada de Programas
- Comunicación entre usuarios y con otras computadoras

- Administración de pedidos de usuarios para usar programas y espacio de almacenamiento.

Las aplicaciones se programan para que funcionen encima de un sistema operativo particular, por tanto, la elección del sistema operativo determina en gran medida las aplicaciones que se pueden utilizar.

Existen infinidad de **S.O. enfocados a diversas aplicaciones**. Con ello puede configurarse la Raspberry Pi como un media center, una videoconsola o centrar su funcionamiento en la programación.

Entre los **principales sistemas operativos** destacan las diversas **distribuciones que ofrece Linux** y la reciente distribución **Windows 10 IOT**. Sin embargo, si lo que queremos es emplear la Raspberry como Media Center, las distribuciones más empleadas son la OpenELEC o RaspBMC.

En este caso, y puesto que se va a trabajar con la Raspberry para programación, el estudio del sistema operativo se centrará en las 2 primeras opciones, distribuciones Linux o Windows 10 IOT. Dentro de las distribuciones de Linux, la más empleada y conocida es Raspbian, S.O. original, realizado por la Fundación Raspberry Pi.

2.3.2. Sistemas operativos

Antes de centrarse en ellos, se va a realizar una pequeña descripción de los sistemas operativos más empleados y conocidos para las diversas aplicaciones que permite la Raspberry.

Tabla 3. Sistemas operativos (S.O.)

Arquitectura	Uso	Oficial	No oficial
GNU/Linux	PC o servidor	Raspbian	ARCH LINUX
		Pidora	Occidentalis
			HypriotOS
			MOEBIUS
			Minibian
			VoidLinux
			OpenMediaVault
			Kano
			OpenSUSE
			pipaOS
			PiBang
			LinuTOP
			Q4OS
			DietPi
	Media CenterXBMC	OSMC	XBIAN
		OPENELEC	
	Solo Raspberry Pi	Ubuntu Snappy Core	Ubuntu 14,04
			Ubuntu MATE
			Gentoo
	Emuladores		Recalbox
			Retropie

Linux	Android o similar		Android NO FUNCIONAL
			Android Lollipop NO FUNCIONAL
			Tizen
			SailPi

No Linux	PC	RISC OS	FreeBSD
			NetBSD
			Windows 10 IoT
			Plan9
			Inferno OS
			Inferno Pi
			WXinuPi

2.3.2.1. S.O. para uso genérico

- **Raspbian OS**

Raspbian OS es la **distribución por excelencia** para la Raspberry Pi. Cuenta con apoyo oficial, y se trata de la opción más completa y optimizada. Está **basado en la distribución Debian Wheezy** (Debian 7.0) optimizando el código de ésta para la placa Raspberry.

Se mueve ágilmente por el hardware de la Raspberry gracias a su **ligereza**. Además hay que destacar su **entorno de escritorio** LXDE, su navegador predeterinado Midori y las **herramientas de desarrollo** preinstaladas que incluye, como el interprese IDLE para Python, Scatch para programar videojuegos, la tienda oficial Pi Store, etc.

- **Kano OS**

Kano OS es una **distribución Linux** especialmente pensada **para** los **niños**. Destacan la multitud de posibilidades que ofrece a los más pequeños y tiene especial interes en el **campo de la educación** debido a sus peculiaridades. Aunque Kano OS **se basa en Debian**, se ha **simplificado** al máximo para enseñar las virtudes de la tecnología a edades tempranas.

- **pipaOS**

pipaOS es otra distribución que **se basa en Debian Wheezy** y especialmente pensada para la Raspberry Pi. Lo que más **destaca es su ligereza**, consiguiendo un tiempo de arranque inferior a los 10 segundos, todo un reto si se tiene en cuenta el limitado hardware de la Pi. Por lo tanto es una distribución que destaca por su **rapidez** e **interactuactuación con otros elementos** como dispositivos USB y smartphones o tablets.

Ocupa muy poco espacio, con unos 420MB el sistema puede funcionar perfectamente. Además cuenta con soporte integrado para los dispositivos USB inalámbricos más populares, rapidez garantizada, sincronización de hora, personalizable, transmisión de radio FM, etc…

- **Ubuntu MATE**

La famosa distribución Ubuntu, pone a disposición Ubuntu MATE para la Raspberry PI 2. **Se trata de un Ubuntu Linux** con un **entorno de escritorio MATE**, ligero y que necesita de pocos recursos. MATE se basa en GNOME2 y es muy conocido en el mundo Linux por su gran aceptación dentro de la comunidad, tanto es así, que existen numerosas distribuciones que lo han elegido como escritorio por defecto.

Con esta distribución se obtiene un sistema UBUNTU completo para realizar gran variedad de cosas. Se trata de una distribución potente, por lo que exige disponer al menos de la Raspberry Pi 2.

- **Tizen**

Tizen es un sistema operativo **pensado para dispositivos móviles**, **basado en Linux** y **de código abierto**, por lo que compite con Android y Firefox OS. Se trata de un un sistema basado en HTML5 (por lo que si se sabe programar en este lenguaje, se puede crear extraordinarias aplicaciones para Tizen) y con muchas similitudes con Firefox OS.

Tizen está patrocinado por la Linux Foundation y tiene apoyo de Samsung, además de la colaboración en el desarrollo de otras empresas como Intel, Huawei, Fujitsu, NEC, Panasonic, Orange, Vodafone, etc. Y aunque parece que Android ha tapado a esta distribución hasta el momento, pronto se hablará mucho de ella.

- **Android Pi (Razdroid)**

Puesto que Android es un **sistema operativo que Google ha creado para la plataforma ARM** (arquitectura para ordenadores con conjunto reducido de instrucciones, empleada en microprocesdores y microcontroladores, como puede ser la Raspberry, los móviles o tablets), es lógico que pueda funcionar en la Raspberry Pi. Sin embargo, se trata de una comunidad de voluntarios que dan soporte a este sistema para la Raspberry. Se ha realizado un gran trabajo en cuanto a la adaptación y optimización de este sistema operativo en la plataforma Raspberry, para que se pueda disfrutar de una gran experiencia en Andorid.

2.3.2.2. S.O. especializados

- **Minibian**

Minibian proviene de **MINImal raspBIAN**, es decir, se trata de una **imagen de Raspbian modificada para ser ultraligera**. Con ello se ha conseguido que la velocidad de empleo sea mayor que la de Raspbian, además de basarse siempre en la última versión de éste, y es totalmente compatible con todas los modelos de Raspberry. A cambio de ser tan ligera, Minibian **no cuenta con interfaz gráfica**, por lo que va a tocar trabajar desde la línea de comandos, resultando más complejo para los novatos. Pero esta elección hace que arranque en solo 10 segundos y solo utilice 29MB de RAM, ocupando tan solo 450MB en la SD.

- **W10 IoT Core (Windows 10)**

Microsoft también ha apostado por la plataforma ARM y se ha adaptado para funcionar en la Raspberry. Con su inminente llegada, parece que están dispuestos a apostar por la Raspberry Pi poniendo a disposición de la comunidad su último sistema operativo. Con ello se garantizan que habrá más desarrolladores de aplicaciones para su nueva plataforma.

2.3.2.3. Emuladores de consolas

- **RetroPie**

RetroPie es una **distribución Linux que se emplea para emular consolas.** Con ella se podrá transformar la Raspberry Pi en un centro de videojuegos y entretenimiento. Está basada en Raspbian y se puede controlar mediante distintos mandos y controladores de consolas y ordenadores.

Entre las consolas o sistemas soportados están Amiga, Atari (800, 2600, ST,…), C64, Final Burn Alpha, Game Boy, Game Boy Advance, Game Boy Color, Sega, MAME, MSX, PC (juegos de PC de 32 bits x86), NeoGeo, Nintendo, TurboGRafx 16, Sinclair ZX, Play Station 1, etc. Como ves, todo consolas vintage para disfrutar de los clásicos en tu Raspberry.

2.3.2.4. Media Centers

- **OpenELEC**

OpenELEC es un sistema operativo oficial de Raspberry Pi, basado en Linux, y su función está destinada a la creación de un **centro multimedia** propio y barato con la Raspberry Pi. Se dispondrá de todo el contenido multimedia y acceso a Internet para transformar la TV en una auténtica smartTV y un centro de ocio sin igual.

Para ello, OpenELEC se basa en el famoso **Kodi** (anteriormente conocido como XBMC, siglas de Xbox Media Center), que es un centro multimedia que fue creado en un inicio para la videoconsola Xbox, pero el desarrollo hizo que se moviese a otras plataformas. Se completa con reproductores de audio, vídeo, presentación de diapositivas, visores de imágenes, reportes de clima, y otras funciones implementadas mediante plug-ins, etc.

2.3.2.5. La Nube y Redes

- **arkOS**

arkOS es un sistema operativo cuyo objetivo principal es **disponer de un hosting doméstico para alojar webs propias o servicios en la red,** como servidores de ficheros, emails, etc… Se puede decir que permite crear una **nube personal** y barata.

Se basa en Arch Linux y se ha personalizado para trabajar de forma óptima con la Raspberry Pi. Además su configuración es sencilla, puesto que todo se puede hacer desde su interfaz gráfica Genesis.

2.3.2.6. Distribuciones para usos específicos

- **OpenDomo OS**

OpenDomo OS es un sistema operativo de código abierto especialmente **diseñado para domótica**. Con él se podrá crear una vivienda totalmente domotizada, además de crear automatismos independientes, realizar proyectos para ahorro energético, etc. OpenDomo **está basado en Linux**.

OpenDomo OS está disponible tanto para Raspberry Pi como para ejecutarlo en una máquina virtual desde un PC para practicar la domótica.

- **OctoPrint**

OctoPrint es un **software gratuito de código abierto**, desarrollado por BQ, diseñado para la Raspberry, cuyo objetivo final es el **control de impresoras 3D**. Tiene una interfaz muy sencilla que permite acceder a todas las funciones de la impresora 3D desde el navegador web, permitiendo interactuar mediante códigos, archivos GCode y realizando reconfiguraciones a la impresora sabiendo que todo va a quedar guardado en OctoPrint para la próxima vez que lo se quiera utilizar. Una de las ventajas de Octoprint es que se puede ver la impresión vía Internet utilizando una webcam, vigilando y controlando en todo momento el estado de la impresión.

2.3.2.7. Extras

- **NOOBS**

NOOBS **no es un sistema operativo**, **es una imagen ISO** que **permite disponer de los sistemas operativos oficiales** una vez instalada en la SD. Su sistema de inicio te permite seleccionar el sistema a ejecutar, pudiendo probar distintos S.O. de forma sencilla y sin tener que estar desinstalando e instalando nuevos sistemas operativos en la tarjeta SD.

NOOBS incluye todos los sistemas operativos oficiales: Raspbian OS, OpenELEC, Snappy Ubuntu Core, Raspbmc, Pidora, RISC OS y Windows 10 IoT. Así que será la mejor opción si se necesita usar más de un sistema operativo.

- **BerryBoot**

BerryBoot **permite instalar varios sistemas operativos** en la Raspberry Pi. Por tanto es un proyecto que compite con NOOBS. Basta con copiar este software en una SD y así se instalará un gestor de arranque que permite tener varios sistemas operativos disponibles en la Rspberry.

Frente a NOOBS presenta inconvenientes y ventajas. Entre los **inconvenientes** encontramos la **configuración del WiFi**, que aunque es sencilla, será necesario realizarla por partida doble,

una en el instalador del BerryBoot y otra en el sistema operativo o distribución que instalemos. Pero quizás la peor desventaja sea que **no instala la última versión del sistema operativo** que queremos, por lo que será necesario realizar una actialización. A pesar de esto, **presenta ventajas**:

- Ocupa solo 30MB en vez del 1GB de NOOBS.
- Permite instalar muchos más sistemas operativos y distribuciones, no solo las oficiales como NOOBS.
- Soporta también la instalación de sistemas operativos desde memorias USB o desde la red (incluso remoto por VNC), pero siempre se necesitará la SD con BerryBoot insertada.
- Reconoce mandos a distancia de la TV por CEC.
- Permite clonar las particiones de los sistemas instalados para poder tener una estable o un backup y otra que utilicemos para nuestros experimentos. Por si algo sale mal.

Tras realizar una breve descripción de los S.O. más importantes y empleados, se va a realizar un estudio más completo de los 2 sistemas más empleados, Rapbian y Windows 10 IoT.

2.3.3. Raspbian

Raspbian es un sistema operativo libre y gratuito **basado en Debian** y optimizado para el hardware de la Raspberry Pi. Raspbian es algo más que un sistema operativo, pues viene con unos 35 mil paquetes, precompilados, de tal forma que sea fácil instalar el que se necesite en cada momento.

Se creó en junio de 2012 por un grupo de fans de la Raspberry Pi, sin embargo está **bajo un desarrollo continuo**, para la mejora de la estabilidad y de los paquetes de Debian. Se trata de la distribución por excelencia para la Raspberry Pi. **Es la más completa y optimizada** de las existentes, por eso cuenta con **apoyo oficial**. Fue el primer sistema y probablemente sea el más extendido para propósitos genéricos.

Raspbian es una **distribución del sistema operativo GNU/Linux** y está basado en Debian Wheezy(Debian 7.0) , al que se le han añadido cambios para adaptarla a las limitaciones de Raspberry Pi, orientándola de esta manera a la enseñanza de informática.

La distribución usa **LXDE como escritorio** y Midori como navegador web. Además contiene **herramientas de desarrollo como IDLE** para el lenguaje de programación **Python** o Scratch, y diferentes ejemplos de juegos usando los módulos Pygame.

Destaca también el **menú "raspi-config"** que permite configurar el sistema operativo sin tener que modificar archivos de configuración manualmente. Entre sus funciones, permite expandir la partición *root* para que ocupe toda la tarjeta de memoria, configurar el teclado, cambiar de contraseñas, habilitar la cámara, etc. Gracias al uso de LXDE como escritorio se puede acceder directamente al menú de configuración a través de éste.

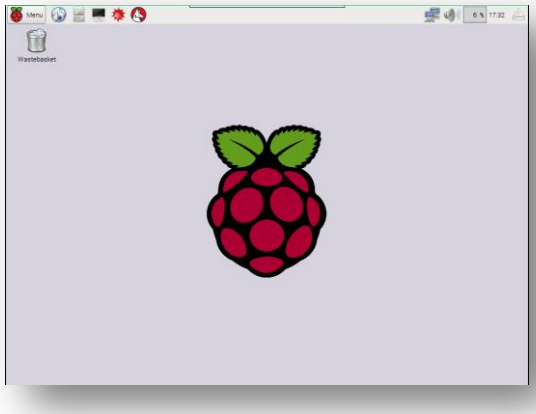


Figura 2. Escritorio del interfaz gráfico de Raspbian

2.3.4. Windows 10 IoT

Windows 10 IoT para Raspberry Pi 2 es solo un **servidor sin interfaz gráfico ni posibilidad de instalar ningún software del Windows normal**.

Las principales **novedades** de este sistema operativo, concretamente de la última compilación lista para Raspberry Pi 2, son:

- Soporte prácticamente total para Python.
- Mejora del rendimiento de los puertos GPIO de la Raspberry Pi 2, de 8 a 10 veces.
- Posibilidad de trabajar a la vez con las tecnologías ADC y PWM.
- Nueva aplicación universal UWP para controlar el sistema operativo.

Universal Windows Platform (UWP) y Hardware:

Facilita la **integración de la interacción entre programas informáticos y el usuario**, **la búsqueda, el almacenamiento online y los servicios en la nube**.

Figura 3. Universal Windows Platform (UWP)

UWP API permite escribir aplicaciones y usarlas en los dispositivos, móviles o escritorios, y da acceso a cientos de dispositivos Windows para usar en los proyectos.

Entre los **dispositivos y aplicaciones** a los que da acceso cabe destacar 3:

- **Arduino Wiring API**

Figura 4. Arduino Wiring API

Windows 10 IoT Core también facilita el uso de Arduino Wiring API, facilitando sus esquemas y librerías para permitir el uso directo de su hardware.

- **Visual Studio**

Figura 5. Visual Studio

Para desarrollar las aplicaciones, Windows pone a disposición del usuario el programa Visual Studio Community Edition. Visual Studio es una herramienta de desarrollo de carácter profesional que incluye plantillas de aplicaciones, un potente debugger (depurador), el apoyo de un rico lenguaje de programación, etc.

Sin este programa se dificulta la programación y la instalación de aplicaciones puesto que Windows 10 IoT no cuenta con escritorio ni con sistemas preinstalados de programación.

- **Connect the Dots**

Figura 6. Connect the Dots

Facilita la conexión de los dispositivos con Windows Azure, el cual permite implementar grandes soluciones mediante el aprovechamiento de un avanzado servicio analítico.

Windows IoT Core provee al usuario de una **amplia plataforma** en la que crear pequeños **dispositivos industriales** con la seguridad a nivel empresarial, administración y servicio similares al resto de las ediciones de Windows 10.

Los **requerimientos hardware** dependen de si se compila en un modo *headed* o *headless*. Los dispositicos *headed* tienen visuaización de vídeo y usan un subsitema de vídeo Windows para direccionarlo. Los dispositivos *headless* no poseen visualizador.

Memoria

Headless

256 MB RAM / 2 GB almacenamiento

Headed

512 MB RAM / 2 GB almacenamiento

Procesador

400 MHz como mínimo.

Windows 10 IoT es **compatible con un gran número de lenguajes de programación** por defecto tales como C++, C#, JavaScript y Visual Basic, aunque en el futuro es posible que la API UWP incluya más lenguajes y características universales para el sistema operativo.

Como **gran desventaja** frente a los S.O. basados en Linux hay que destacar que **no posee interfaz gráfico**, y es **menos flexible**.

No se cuenta con un escritorio en Windows 10, sino una **plataforma de desarrollo para todo tipo de proyectos** en ámbitos como la robótica, la domótica, o desde luego la Internet de las Cosas.

Windows 10 IoT Core no es una plataforma que permita abrir un navegador ligero, editar documentos de Office en nuestra placa o escuchar música. La idea con esta plataforma es la de **invitar a los desarrolladores a crear e inventar todo tipo de soluciones adaptadas a la Internet de las Cosas** (IoT) y no centradas en el segmento de los PCs y portátiles tradicionales.

Como ya se ha comentado previamente, no se puede realizar nada mediante la interfaz gráfica de este SO, por lo que requiere del **uso de un SSH**, y tampoco se puede emplear un acceso remoto, puesto que este se basa en la interfaz gráfica del SO.

Para lograr conectar la Raspberry Pi 2 con Windows 10 IoT Core se utulizará **Powershell**, la **consola avanzada que se incluye en Windows 10** y que permite abrir sesiones remotas vía SSH.

Para la creación de aplicaciones se emplea principalmente el programa **Visual Studio 2015**, desarrollado por Windows y enfocado principalmente para el uso de la Raspberry Pi.

La **instalación** de Windows 10 IoT se ha simplificado mucho, ya que aparece en la **última versión de NOOBS**, ya que se trata de un SO oficial, por tanto aparecerá junto a raspbian, openelec, pidora, etc. Bastará con seleccionarlo en la pantalla inicial de NOOBS, y se realizará la instalación automática.

Figura 7. Pantalla de configuración de idioma de Windows 10 IoT

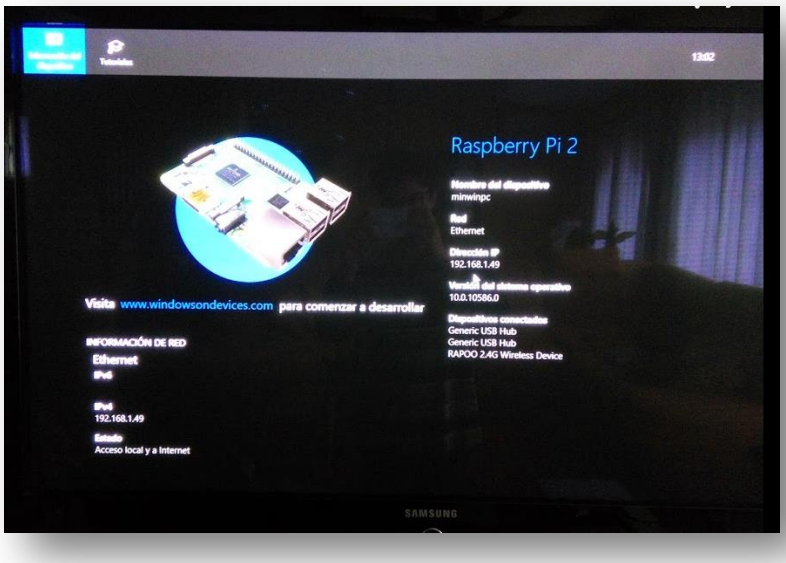


Figura 8. Pantalla general de Windows 10 IoT

En la figura 8 se muestra la pantalla general de Windows 10 IoT, la cual ofrece información acerca del dispositivo y conexiones, y la única interacción que esta permite con el usuario es el obtener tutoriales sobre proyectos realizados en este S.O.

2.3.5. NOOBS

NOOBS (New Out Of Box Software), es una aplicación que **facilita la instalación** de diversas distribuciones Linux, e incluso de Windows 10 IoT, en la Raspberry Pi.

Se trata de una aplicación que contiene los datos necesarios para la instalación de S.O. y da la opción de instalar soluciones como Raspbian, Arch Linux, RaspBMC, el recién salido Pidora u OpenELEC sin problemas, y en la última versión, incluso Windows 10 IoT. La instalación permitirá más tarde iniciar la Raspberry Pi de forma normal con esa nueva distribución, pero NOOBS quedará residente en memoria y permitiendo acceder a esta aplicación siempre que se quiera mediante la pulsación de la tecla Shift durante el proceso de arranque.

Esto permitirá **instalar más de una configuración** en la microSD, además de cambiar a otro S.O. si es necesario.

En lo referente a la instalación, NOOBS instala la **versión actualizada de los S.O.** por lo que no hará falta la posterior actualización del sistema una vez se haya instalado este.

En deinitiva NOOBS es la aplicación recomendada para usuarios con poca experiencia en el campo de la Raspberry Pi, que tienen dudas sobre qué S.O. desean usar.

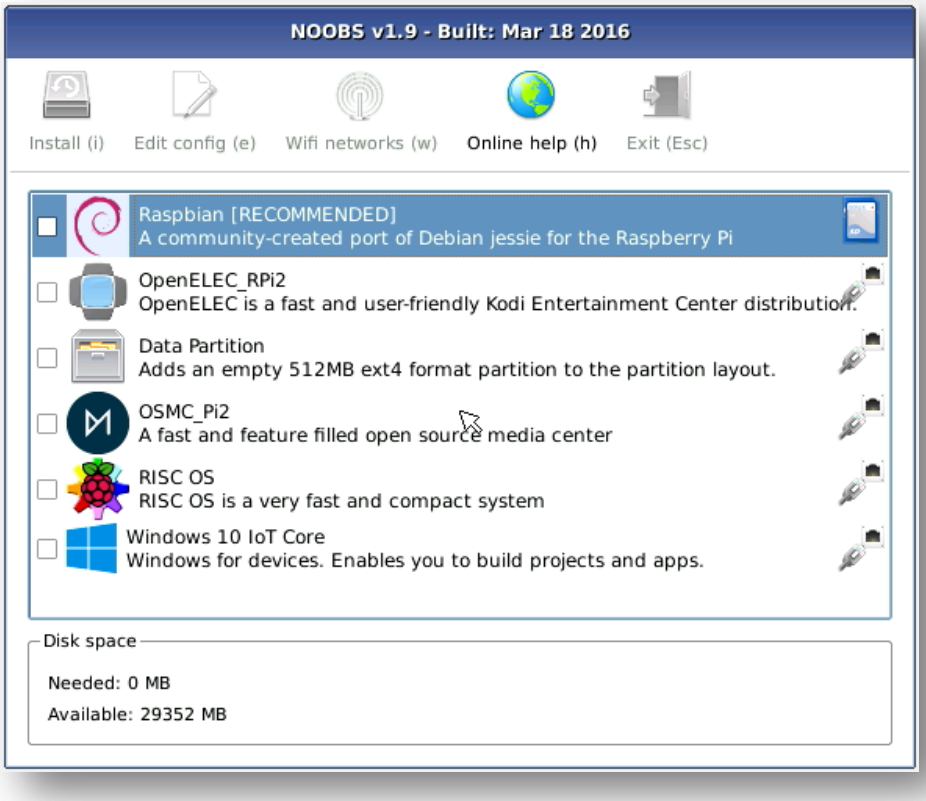


Figura 9. Menú de instalación de NOOBS.

2.3.6. Elección del sistema operativo.

Se ha elegido **Raspbian** como SO de este proyecto frente a Windows 10 IoT debido a:

- Aparece mucha **más información** acerca de Raspbian que de Windows.
- Da la posibilidad de emplear una **interfaz gráfica**, por lo que se podrán realizar los proyectos y aplicaciones a través de un monitor y un teclado, sin falta de un acceso remoto o un SSH. Además da la posibilidad de emplear dicho acceso remoto para la visualización de la interfaz gráfica, sin necesidad de emplear un monitor.
- La **alta interacción** con el S.O., permite navegar por internet, realizar juegos, acceder a distintos sistemas de programación, como puede ser el scrach, python, c+, java…
- Estos **sistemas de programación** vienen **preinstalados** en el S.O., y por tanto se podrán emplear con facilidad una vez configurado el S.O.

Con todo esto, se pueden resumir las ventajas de Raspbian en una, la **creación de un sistema sencillo** para usuarios nuevos, en los que **se facilita el acceso a sistemas de programación y a aplicaciones preinstaladas** que hacen a Raspbian asemejarse a un ordenador , resultando más sencillo de utilización para usuarios que no han trabajado con esta placa anteriormente.

Puesto que se quiere realizar un programa de monitorización de fácil empleo por el usuario, se va a emplear el Raspbian, que simplifica la interacción con la Raspberry gracias a la interfaz gráfica, y va a simplificar la programación y configuración inicial.

Sin embargo, **se va a instalar NOOBS** en lugar de Raspbian directamente, ya que va a dar la opción de preinstalar ambos S.O., Windows 10 IoT y Raspbian, pudiendo de esta manera probar los dos SO sin necesidad de borrar los cambios realizados en cualquiera de ellos ni de tener que realizar la instalación de los mismos antes de usarlos.

A su vez, NOOBS va a simplificar la instalación de cualquier S.O. y de él mismo, ya que simplemente habrá que grabar la imagen de NOOBS en la tarjeta SD para comenzar a usarlo.

2.4. Herramientas

En este apartado se va a realizar una breve descripción de los **programas necesarios para la instalación del SO y la programación de aplicaciones**.

Para la instalación de ciertos S.O. en la tarjeta SD se requiere de un programa capaz de volcar y grabar la imagen del S.O. en la microSD.

La programación de la Raspberry Pi se puede realizar de dos maneras, mediante el uso de programas instalados en la interfaz gráfica, o mediante el uso de programas de acceso remoto a través de un ordenador conectado a la misma red que la Raspberry.

Éstos últimos serán de gran importancia en el caso de SO sin interfaz gráfica, puesto que se tratarán de la única forma de programación posible.

2.4.1. Visual Studio

Se trata de un **entorno de desarrollo** integrado para crear aplicaciones espectaculares para Windows, Android e iOS, además de aplicaciones web y servicios en la nube innovadores.

Permite escribir código en C#, Visual Basic, F#, C++, HTML, JavaScript, Python y mucho más.

Posee **herramientas de codificación** eficaces con la que se puede escribir código y buscar y corregir problemas de código rápidamente.

También dispone de **herramientas web** para crear aplicaciones web modernas basadas en ASP.NET, Node.js, Python y JavaScript. Se usa con marcos web potentes como AngularJS, jQuery, Bootstrap, Django y Backbone.js.

Con esto se pueden crear aplicaciones que se comuniquen entre estaciones de trabajo, páginas web, dispositivos móviles, dispositivos embebidos, consolas, etc.

Se trata del SSH definido por excelencia para **Windows 10 IoT**, por tanto se va a dar una breve explicación sobre como conectarlo con este S.O.

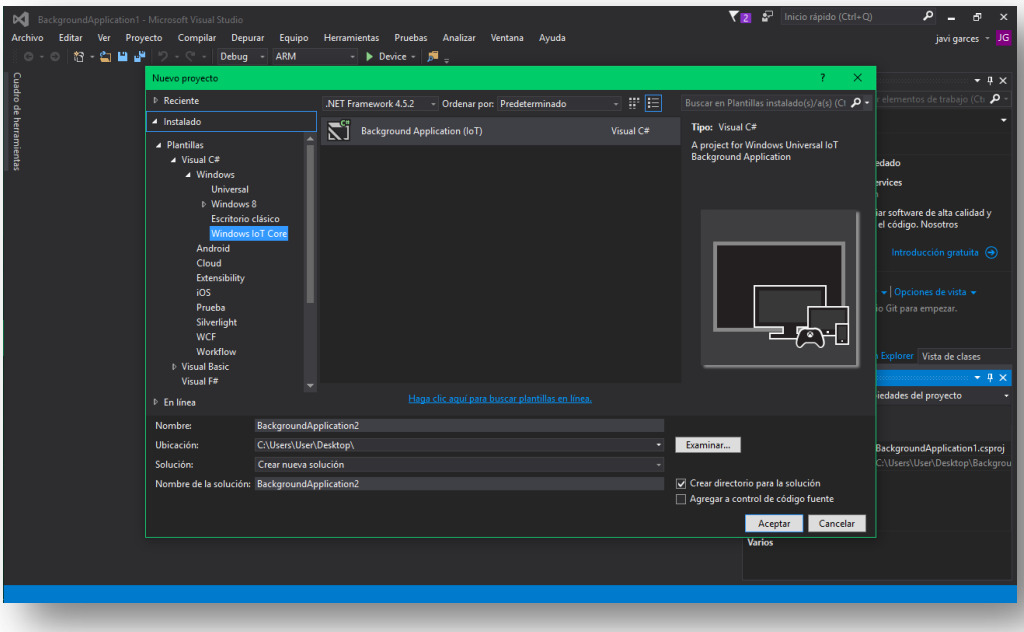


Figura 10 Configuración de un nuevo proyecto en Visual Studio.

La **conexión con la Raspberry Pi** es muy sencilla, solamente hace falta descargar el archivo de visualización Windows IoT core, conectarlo con el dispositivo a través de la red local y abrir un nuevo proyecto:

Como ya se ha comentado anteriormente, se trata de un SSH, por lo que solo se podrá escribir comandos en Windows 10 IoT a través de esta aplicación.

2.4.2. PuTTY

PuTTY es un **emulador de terminal**, un programa que permite **conectarse con equipos remotos** y ejecutar programas a distancia. PuTTY **se conecta como cliente** a múltiples protocolos, como SSH, Telnet o Rlogin.

Para este proyecto, se empleará la conexión como SSH a través de la **red local** y el **puerto 22 de la Raspberry**.

Se ha elegido este programa debido a su **sencillez** y **facilidad para la conexión y la programación**, puesto que con conocer la dirección de la Raspberry Pi, el nombre de usuario y la contraseña de la misma, se podrá programar la Raspberry Pi desde el ordenador remoto.

Para la utilización de este programa, una vez descargado, basta con ejecutarlo para obtener la pantalla de inicio de PuTTY.

Antes de ejecutarlo, hay que asegurarse de que tanto la Raspberry Pi como el dispositivo con el programa PuTTY, deben estar en la misma red local.

En la pantalla de inicio de PuTTY habrá que introducir la dirección IP de la Raspberry, y el puerto de ésta que está configurado como enlace entre la misma y el servidor PuTTY. Una vez elegido SSH como tipo de configuración, se pulsa el botón Open para acceder a la pantalla de la figura 12.

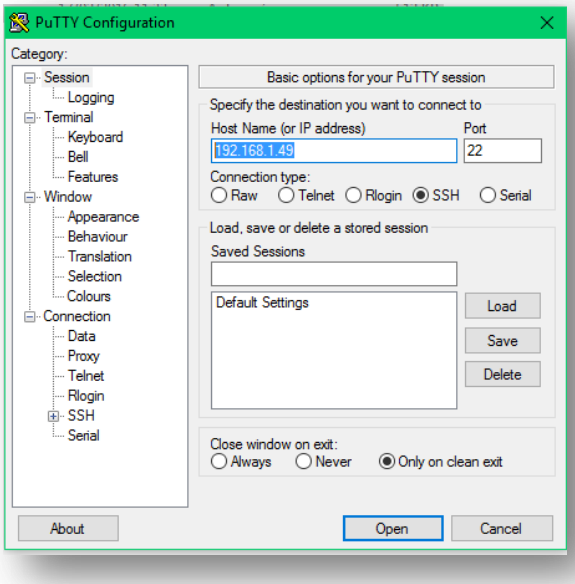


Figura 11. Ventana de configuración de PuTTy.

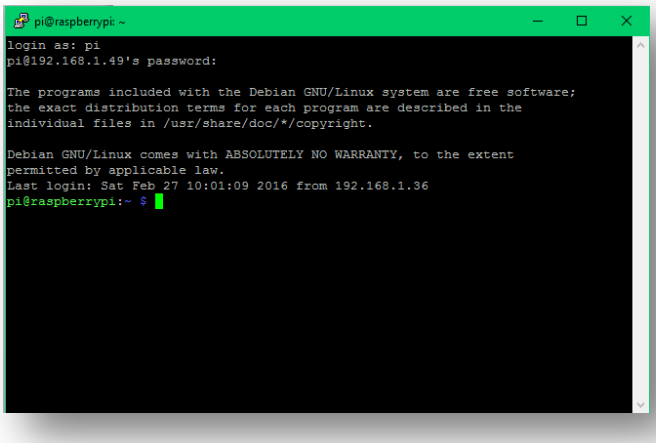


Figura 12. Terminal obtenido al emplear PuTTy.

Esta pantalla representa la línea de comandos de RaspBian, la cual va a pedir el nombre de usuario y contraseña de la Raspberry. Tras introducirlos correctamente, aparecerán el mensaje

de la imagen anterior y la frase “pi@raspberrypi:- $”, que permitirá introducir los comandos y ejecutarlos en a Raspberry Pi.

Se podrá descargar este programa desde:

http://www.putty.org/

2.4.3. VNC viewer

Virtual Network Computing (Computación Virtual en Red).

VNC viewer es un **sistema de visualización remota**, que permite el **intercambio y control de escritorio gráfico** entre dos dispositivos. También es conocido como **software de escritorio remoto**.

A su vez transmite el control del ratón y teclado, y recibe las actualizaciones de la pantalla a través de la red desde el servidor remoto. Se conseguirá visualizar el escritorio de RaspBian en una pestaña de nuestro ordenador una vez ejecutado el programa.

Para configurar VNC viewer hay que **descargar** el programa en el dispositivo receptor, e instalar ThighVNC server en la Raspberry Pi.

Para realizar la **instalación** de ThighVNC server hay que introducir la siguiente línea de comandos, empleando para ello el terminal de RaspBian o un servidos SSH:

```
sudo apt-get install tightvncserver
```

Con ello se instalarán los paquetes de ThighVNC en la Raspberry. A continuación hace falta **iniciar el servicio de VNC** server introduciendo el siguiente comando:

```
vncserver :1 –geometry 1280x800
```

Con la introducción de vncserver :1 se indica el número de escritorio que está corriendo, y hará falta para acceder remotamente al escritorio. El segundo término, geometry, indica el tamaño de la pantalla en pixeles, y se podrá ajustar para las dimensiones dl monitor.

La primera vez que se ejecuta el programa pedirá la introducción de una contraseña de acceso, y otra de configuraración VNC en modo solo lectura.

Una vez instalado y ejecutado VNC en la Raspberry hay que configurar el visualizador de VNC en el otro dispositivo con el que se accede remotamente. Para ello hay se empleará **Real VNC Viewer**. Una vez se ha descargado el programa, se podrá ejecutar (no requiere de instalación), y aparecerá una ventana como la siguiente:

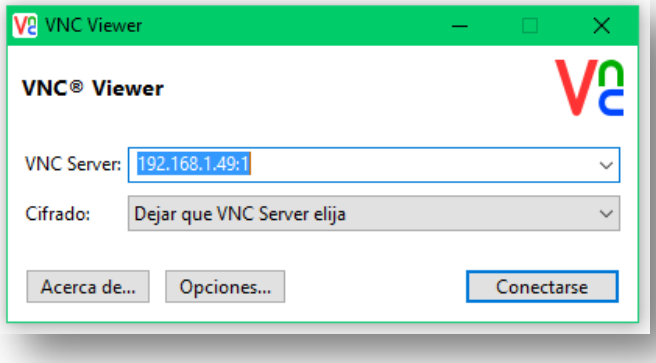


Figura 13. Ventana de acceso a VNC Viewer.

En ella se introducirá la dirección IP de la Raspberry seguida de dos puntos y del número de escritorio que se haya ejecutado para VNCserver en el terminal de ésta. Una vez pulsado el botón “Conectarse” se pedirá la contraseña configurada anteriormente.

Una vez introducida la contraseña y aceptado, emergerá una ventana cuyo contenido es el escritorio gráfico desde el que se podrá controlar la Raspberry Pi.

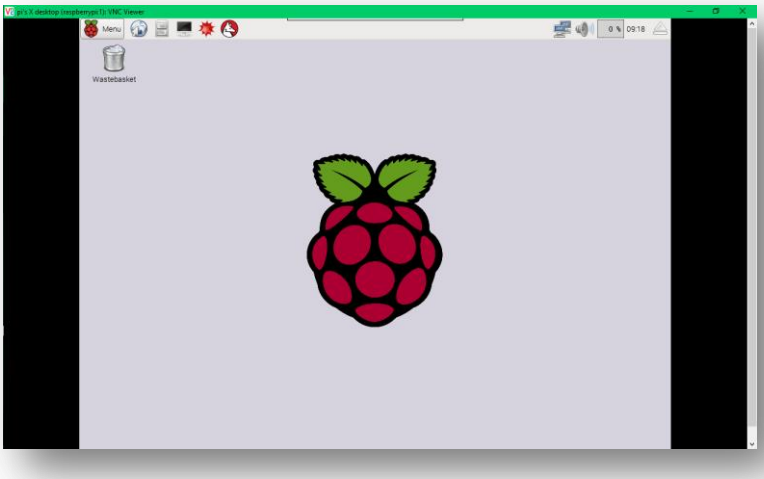


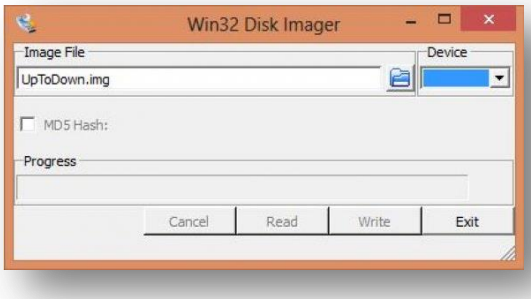
Figura 14. Interfaz gráfico obtenido con el empleo de VNC Viewer.

Podrá descargarse tanto la versión ejecutable como la instalable de este programa desde:

https://www.realvnc.com/download/viewer/

2.4.4. Win32 Disk Imager

Win32 Disk Imager es una sencilla aplicación de código abierto que **graba imágenes de CD o DVD en una memoria USB o en una tarjeta SD**, creando un **lector de discos virtual**.



Sólo hay que descomprimir el programa (no requiere instalación) y seleccionar en la tarjeta SD la imagen que se quiere grabar, y a continuación el dispositivo en el que se quiere grabar. Con esto la tarjeta de memoria se convierte en un CD virtual, capaz de ser usado por la Raspberry.

Figura 15.Win32 Disk Imager

Este programa se puede descargar desde:

http://win32-disk-imager.uptodown.com/windows

2.4.5. SDFormatter

SD Formatter es un software que proporciona un **acceso rápido y sencillo a todos los formatos de tarjetas de memoria SD**, SDHC y SCXC, y con el cual se puede **eliminar** de una sola vez el **contenido** almacenado en la memoria SD.

Para formatear la tarjeta de memoria SD hay que seguir los siguientes pasos:

Descargar y abrir el programa SDFormatter. Una vez en el programa, seleccionar opciones y configurar los siguientes parámetros:

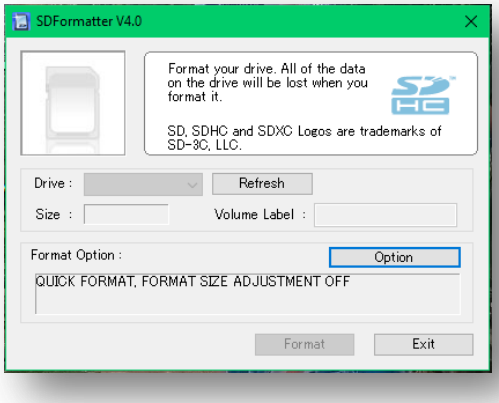


Figura 16. Ventana inical de SDFormatter.

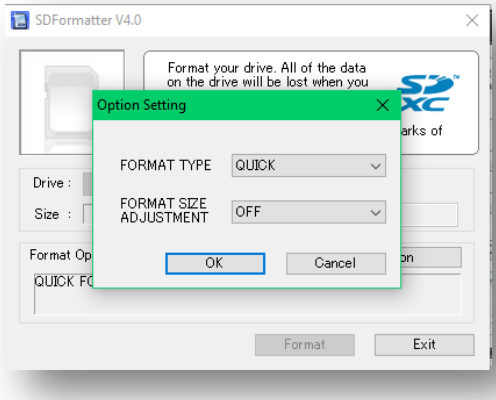


Figura 17. Ventana de ajustes de SDFormatter.

Si se selecciona FORMAT SIZE ADJUSTEMENT OFF hay que introducir el tamaño de la tarjeta manualmente, sino se selecciona ON y el programa ajustará este parámetro al indicado por la tarjeta.

Por ultimo hay que seleccionar la unidad de la tarjeta, y pulsaar el botón Format.

Figura 18. Configuración final de SDFormatter.

Con esto se conseguirá formatear la tarjeta ajustándola a su tamaño original para no perder capacidad.

Se puede descargar este programa desde:

https://www.sdcard.org/downloads/formatter_4/

2.4.6. Advanced IP Scanner

Se trata de un **explorador de redes,** rápido y gratuito, que permite **obtener información sobre los dispositivos de la red**, acceder a las carpetas compartidas y a los servidores FTP, proporciona control remoto de las computadoras (mediante RDP y Radmin) e incluso puede apagar los dispositivos de manera remota. Es fácil de usar y se ejecuta como una edición portátil, por lo que no requiere de instalación.

Este programa no es indispensable para la realización del proyecto, ya que se podrá conocer la dirección IP de la Raspberry a través del terminal de RaspBian, introduciendo el **comando ifconfig.**

Para emplear el programa, simplemente hay que ejecutarlo y pulsar el botón Explorar. Con ello aparecerá una lista con los dispositivos conectados a la misma red que el dispositivo en que se haya abierto este programa.

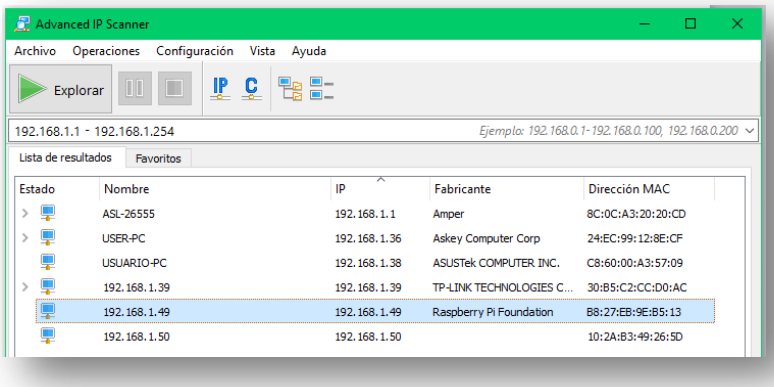


Figura 19. Advanced IP Scanner.

La descarga de este programa se realizará desde:

http://www.advanced-ip-scanner.com/es/

2.5. Periféricos

Se trata de **dispositivos auxiliares e independientes** que se **conectan a la unidad central** de procesamiento de la placa de programación, cuya **función es comunicarse con el exterior** (dispositivos de entrada/salida) o **archivar y almacenar información** de sistema (memorias auxiliares).

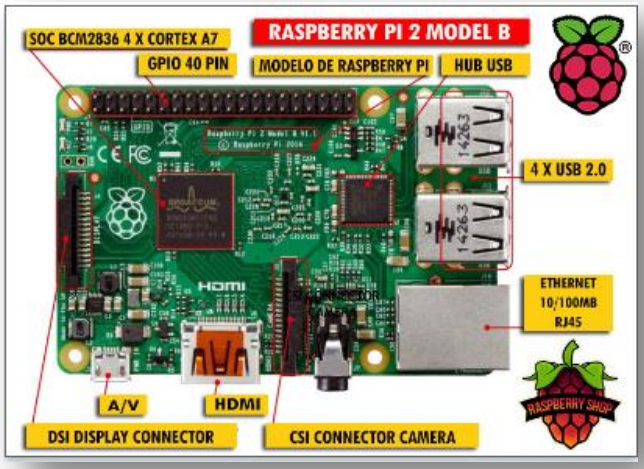


Figura 20. Puertos de conexión de la Raspberry Pi 2B.

En este apartado se va a realizar una breve descripción de los periféricos principales de la Raspberry Pi, así como de los periféricos empleados en este proyecto.

En la figura 20 aparecen los puertos de conexión de la Raspberry Pi 2B:

2.5.1. Teclado y ratón

Estos dos dispositivos se emplean para la **comunicación entre el usuario y el S.O.** Su principal función es la de permitir el acceso y navegación del usuario **en el entorno de trabajo** del S.O. instalado, para la introducción de comandos y para realizar la configuración y la programación de aplicaciones.

Estos periféricos resultan de mayor importancia en los S.O. basados en Linux, ya que poseen un entorno gráfico muy útil desde el cual se puede acceder a todos los archivos y programas del sistema, a la vez que permite modificarlos y crear aplicaciones propias.

Para el uso del ratón y teclado se dispone de un **conector USB** que permite la conexión de estos dispositivos con la rpi.

2.5.2. Cámara

La Raspberry Pi permite la conexión de cámaras como **dispositivos de entrada de datos,** tanto **cámaras USB**, como **cámaras diseñadas explícitamente para la Raspberry**, como es el caso de este proyecto, conectadas directamente al puerto de cámara CSI.

Ya que no todos los modelos de cámaras son compatibles con ciertos modelos de la Raspberry, en el siguiente enlace se puede verificar la compatibilidad de los dispositivos.

http://elinux.org/RPi_USB_Webcams

2.5.3. Salida de video

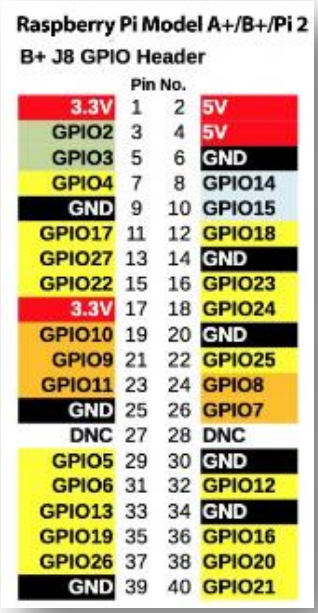
La Raspberry Pi permite conectar dispositivos de **salida de video** entre la misma y un monitor. Dispone de **3 modos** distintos de conexión: **salida de video compuesto**, **conector display DSI**, y **video HDMI**. Este último es el modo de conexión más empleado.

- **HDMI**

La Raspberry posee un **conector HDMI**, que permite una conexión de **alta calidad**, aumentado la resolución de las imágenes y videos mandados, alcanzando calidades de imagen de Full Hd o 1920x1080p.

2.5.4. Puertos GPIO (General Purpose Input/Output)

Se trata de un **bus de expansión de 40 pines** que permite a la Raspberry Pi comunicarse con el exterior tanto para activar elementos como para leer el estado de los mismos. **La tensión de trabajo** del puerto es de **3,3V** para un 1 y **0V** para un 0. Sin embargo posee también pines de tensión 5V y 3.3V para la adaptación de dispositivos con una alimentación de dicha tensión.



La **corriente máxima** que puede suministrar cada pin es de **16 mA**, siendo **50 mA** la **máxima corriente total** suministrada por los puertos GPIO.

Los chips y circuitos de la Raspberry Pi funcionan con 3.3 voltios, por lo que si se conecta un sensor que envíe señal a 5V a través de los GPIO es probable que después la Raspberry Pi no funcione.

Todos los pines GPIO se pueden gestionar directamente a través de código, ya sea para poner un valor o leer un valor de un elemento externo.

Figura 21. Puertos GPIO de la Raspberry Pi 2B.

En este proyecto se van a programar los pines GPIO para controlar el **movimiento de dos servomotores**.

2.5.5. Servomotores

Un **servomotor** o **servo** es un **motor electrónico de baja inercia** al que se le puede **controlar** tanto la **velocidad de giro** como la **posición** dentro de su rango de operación. El cuerpo del servo está formado por un **motor electico**, una **caja reductora con engranajes** y un **circuito electrónico de control,** que sirven como sistema de regulación del motor y como sistema sensor que controla el movimiento del mismo.

Son empleados principalmente en aplicaciones en las que se requiere de un **ajuste preciso en el giro del motor**, como puede ser en sistemas de dirección o en el movimiento de cámaras de seguridad.

Los servos usan la **modulación por ancho de pulso (PWM)** para controlar la posición del motor eléctrico. Su **alimentación** es de **5V**, y requieren de otros 2 pines, uno conectado a **GND**, y otro conectado al pin de **salida PWM**.

Figura 22. Ejemplo servomotor Tower Pro.

2.6. Proyectos realizados anteriormente.

Existen gran variedad de proyectos realizados para la Raspberry. Estos proyectos abarcan funciones muy diferentes, como puede ser la grabación y videovigilancia, o adquisición de datos. Sin embargo se pueden realizar proyectos muy diferentes, como la creación de un espejo de leds, una máquina Arcade, o crear un equipo de música.

Todos estos y más proyectos se pueden encontrar en la web, lanzados en libros oficiales de proyectos. Un ejemplo de ellos es el siguiente:

https://www.raspberrypi.org/magpi-issues/Projects_Book_v1.pdf

También se pueden encontrar foros de aficionados en los que se suben proyectos realizados:

https://www.hackster.io/raspberry-pi

En la página oficial de Raspberry también se podrán encontrar pequeños programas y la explicación sobre estos:

https://www.raspberrypi.org/education/

Otra web muy útil para la obtención de proyectos ya realizados y búsqueda de información es:

https://raspberryparatorpes.net

3. Lenguajes de programación

El lenguaje de programación es un **sistema de comunicación**, con una **determinada estructura, contenido y uso**, para la **creación de un código fuente** de un software. Mediante el lenguaje de programación se van a creas estructuras que, con cierta base sintáctica y semántica, impartirán las instrucciones a un programa de ordenador.

Los lenguajes de programación más empleados en la Raspberry para el S.O. Raspbian son **Python, PHP y el propio lenguaje del terminal**.

3.1. Python

Python en un **lenguaje de programación de propósito general**, orientado a objetos, que también puede utilizarse para el desarrollo web. Es un lenguaje de **alto nivel**, con un enfoque simple pero efectivo que funciona en la mayoría de las plataformas. Además su filosofía hace hincapié en una sintaxis que favorezca un código legible.

El intérprete de Python y la extensa biblioteca estándar están a libre disposición en forma binaria y de código fuente para las principales plataformas desde el sitio web de Python, http://www.python.org/.

Características del lenguaje Python:

- **Propósito general**

Se pueden crear **todo tipo de programas**. No es un lenguaje creado específicamente para la web, aunque entre sus posibilidades sí se encuentra el desarrollo de páginas.

- **Multiplataforma**

Hay versiones disponibles de Python en **muchos sistemas informáticos distintos**. Originalmente se desarrolló para Unix, aunque cualquier sistema es compatible con el lenguaje siempre y cuando exista **un intérprete programado** para él.

- **Interpretado**

Quiere decir que **no se debe compilar el código antes de su ejecución**. En realidad sí que se realiza una compilación, pero esta se realiza de manera transparente para el programador. Esto ofrece ventajas como la **rapidez de desarrollo** e inconvenientes como una **menor velocidad**.

- **Interactivo**

Python dispone de un **intérprete por línea de comandos** en el que se pueden introducir sentencias. Cada sentencia se ejecuta y produce un resultado visible, facilita el entendimiento

del lenguaje y la verificación de los resultados de la ejecución de porciones de código rápidamente.

- **Tipado dinámico**

Un lenguaje de tipado dinámico es aquel **cuyas variables**, **no requieren ser definidas** asignando su tipo de datos, sino que éste, se auto-asigna en tiempo de ejecución, según el valor declarado

- **Orientado a Objetos**

La programación orientada a objetos está soportada en Python y ofrece en muchos casos una manera sencilla de crear programas con componentes reutilizables.

- **Funciones y librerías**

Dispone de muchas funciones incorporadas en el propio lenguaje, para el tratamiento de strings, números, archivos, etc. Además, existen muchas **librerías** que se pueden importar en los programas **para tratar temas específicos** como la programación de ventanas o sistemas en red o cosas tan interesantes como crear archivos comprimidos en .zip.

También se pueden incorporar librerías para el empleo de los **puertos GPIO**, para crear **interfaces gráficos** de interacción con el usuario mediante TKINTER o para emplear la **cámara**.

Para poder ejecutar programs en Python se emplea un **Shell**, que es un **intérprete de comandos basado en texto,** cargado dentro de una terminal.

Sentencias para la programación:

- Bucle **while**:

```
while b < 10:
    comandos
```

- Bucle **if**:

```
if x < 0:
    comandos
elif x == 1:
    comandos
else:
    comandos
```

- Bucle **for**:

```
Cadena= “abcde”
for i in cadena:
        print i
```

- El commando **print** escribirá en el Shell que ejecuta el programa el valor de la variable deseada.

- Mediante el comando **import** se importan las librerías necesarias para ejecutar os programas.

  import RPi.GPIO as GPIO : con este comando se instalarán las librerías de los puertos GPIO empleado posteriormente en el proyecto.

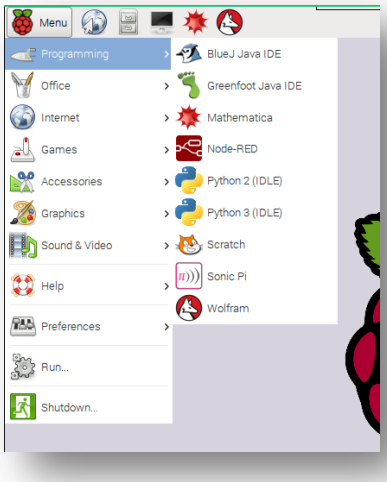
  import time : con esta librería se podrán emplear funciones que utilicen retardos temporales.

- **Definición de funciones**:

  def nombre_de_la_función (variables):

  Dentro de la función se escribirá el comando *return* o *print* para que nos devuelva el valor del resultado que se desee obtener.

Para llamar a la función basta con escribir el nombre de la función en el programa seguida de un paréntesis con el valor que se le quiere dar a la variables.



En cuanto a la Raspberry Pi, el **intérprete** viene **preinstalado en RaspBian**, por lo que el empleo de Python en este S.O. será rápido y sencillo. Este Python Shell se llama **IDLE**, y permite la interacción de manera básica entre el usuario y este lenguaje de programación.

El **acceso** a este intérprete se encuentra en el menú de Raspbian, en programación:

Figura 23. Acceso a IDLE.

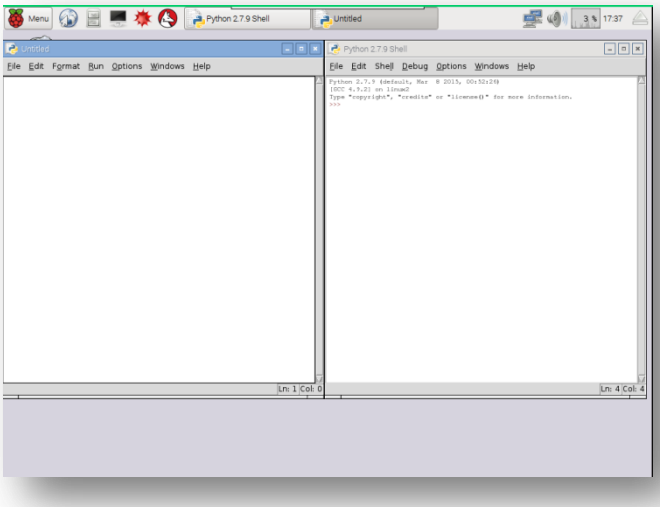


Figura 24. Interprete IDLE para Python.

Al ejecutar el programa se abrirá una ventana con el intérprete. Para abrir un archivo nuevo en el que escribir la programación hay que ir al menú File, y abrir una nuevo fichero.

3.2. PHP

PHP (Hipertext Preprocessor) es un **lenguaje de programación de propósito general**, **interpretado** con una **sintaxis similar a la de C++** o JAVA. Aunque el lenguaje se puede usar para realizar cualquier tipo de programa, es especialmente conocido en el **desarrollo web** ya que puede **incrustarse en páginas HTML** y **es ejecutado por el servidor web**. Por lo que ha alcanzado su máxima popularidad en la **generación dinámica de páginas web.**

Se trata de un lenguaje libre **disponible para muchos SO**, tales como **GNU/Linux**, UNIX o **Windows,** sin embargo, una vez ejecutado en el servidor, el programa en PHP puede ser usado en cualquier tipo de dispositivo y SO. Esto, junto con su **extensa documentación**, ha hecho de PHP un lenguaje muy popular.

Además posee diferentes extensiones que capacitan a PHP para generar documentos PDF, páginas dinámicas, conexión con bases de datos, etc.

En lugar de usar muchos comandos para mostrar HTML, **las páginas de PHP contienen HTML con código incrustado**, encerrado entre las **etiquetas** especiales de comienzo y final **<?php** y **?>** que permiten entrar y salir del "modo PHP".

El código es **ejecutado en el servidor**, **generando HTML** y **enviándolo al cliente**. El cliente recibirá el resultado de ejecutar el script, aunque no se conocerá el código subyacente. El servidor web puede ser configurado incluso para que procese todos los ficheros HTML con PHP.

Una de las **mayores ventajas** de PHP es su gran **sencillez** para el principiante, ofreciendo a su vez configuraciones y características avanzadas para programadores más expertos.

Aunque el desarrollo de PHP está centrado en la programación de scripts del lado del servidor, se puede utilizar para muchas otras cosas. También **pueden ejecutarse los scripts PHP desde la línea de comandos**, sin necesidad de un servidor o navegador.

Se pueden distinguir **tres campos principales** donde se emplear scripts de PHP.

- **Scripts del lado del servidor.**

  Es el **uso más común y popular**. Se trata de la creación de un programa en PHP para la visualización con un navegador, siendo la **página web** ejecutada mediante de un servidor.

- **Scripts desde la línea de comandos.**

  También pueden ejecutarse los scripts PHP desde la **línea de comandos**, sin necesidad de un servidor o navegador, pero es necesario un **analizador de PHP**. En cuanto a S.O.

Linux, bastará con ejecutar los scripts en **cron** (intérprete de PHP en los sistemas Linux). En caso de emplear Windows en el Planificador de tareas.

- **Escribir aplicaciones de escritorio.**

  PHP **no es el lenguaje más apropiado** para la creación de **aplicaciones de escritorio** con interfaz gráfica de usuario. Sin embargo, si se conoce bien este lenguaje se podrán emplear las características avanzadas de PHP para este uso.

Para **incluir código PHP** basta con **encerrarlo entre las etiqueta <?php** y **?>.** Si el servidor web está correctamente configurado, detectará código PHP y, en vez de proporcionar al cliente el contenido de la página directamente, **ejecutará el programa y devolverá su resultado al navegador.**

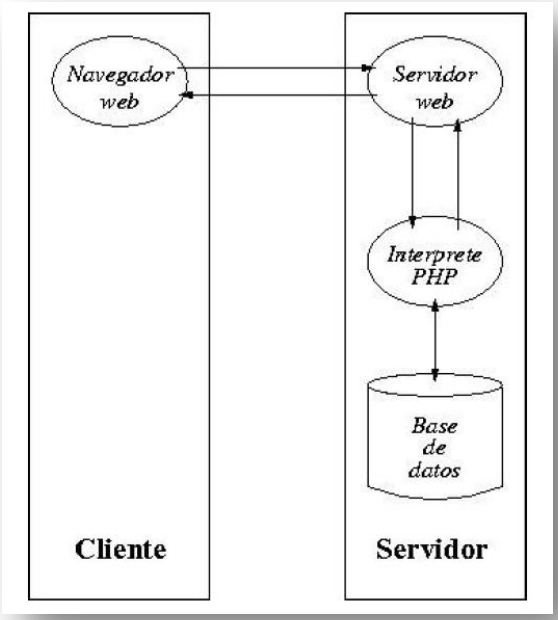


Figura 25. Interacción servidor web - cliente en PHP.
http://servicio.uca.es/softwarelibre/publicaciones/apuntes_php

Para **configurar un servidor web** con **soporte para PHP** en un sistema **GNU/Linux** hay que realizar los siguientes pasos:

- Instalar el **paquete Apache2** con sus dependencias (que contiene el servidor web).
- **Lanzar el servidor** (también conocido como **demonio**) http, invocando al script /etc/init.d/apache2 con el **parámetro start**.

- Probar que Apache sirve peticiones. Abrir un navegador web y escribir la URL localhost (o 127.0.0.1). Deberá mostrar una página de bienvenida como respuesta o decir que no la hay, pero en ningún caso mostrar un error de petición rechazada.
- **Instalar el paquete PHP** (que incluye el lenguaje) **y apache-php** (el paquete que permite conectar Apache con PHP). También se recomienda php-manual, el manual oficial.
- Para comprobar que Apache ejecuta código PHP debe existir el directorio donde Apache busca las páginas web: DocumentRoot /etc/apache2/*.

En cuanto a la instalación en el proyecto, **Apache 2** y sus dependencias, y el **paquete PHP** y **apache-php** quedarán instalados junto con la instalación del programa RPI_WEB_CAM_INTERFACE.

3.2.1. Iniciación a la programación.

Bucles permitidos:

- **For**

  ```
  for ($i = 1; $i <= 10; $i++) {}
  ```

- **If**

  ```
  if ($a < 20) {}
  ```

- **While**

  ```
  while ($i = 1) {}
  ```

- **Do-while**

  ```
  do {
  …
  } while ($i > 0);
  ```

Como se puede observar, la **sintaxis** de los bucles es **similar a la de C**.

En lo referente a las **variables**, hará falta declarar las variables con el **identificador $** delante de la variable:

```
$var
```

Si lo que se quiere es **mostrar un mensaje o variable** en la pantalla, el comando necesario es echo:

```
echo "va es igual a" . $var
```

Si se quiere **mostrar un texto** se escribe **entre comillas** y las variables con el signo $ delante. La conexión entre textos o variables se hace mediante un punto entre ellas.

El programa en PHP se **divide** en diferentes **apartados**:

- En primer lugar, con **<html>** se **inicia el programa que se quiere visualizar mediante el servidor**. Con **</html>** se dará por **concluido** el programa que se quiere mostrar.
- El **encabezado** se podrá entre los comandos <**head> y </head>,** y en él se pueden introducir tanto el **título**, como la **fecha** del programa, etc.
- Por último, se escribirá el **núcleo** del programa entre los comandos **<body> y </body>**. Es en este apartado en el que **se suele iniciar el comando <?php** o delimitador del código PHP, que permite detectar el código PHP, y por tanto la lectura de comandos por el servidor. Con ?> se cerrara la lectura del código PHP.

```
<html>
        <head>
                <title>ejercicio_ejemplo</title>
        </head>
        <body>
                <?php
                        $inicio = "Hola ";
                        $fin = " a todos";
                        echo $inicio.$;
                ?>
        </body>
</html>
```

```
<!DOCTYPE html>
<html>
        <head>
                <title>Ejemplo de una web con PHP - aprenderaprogramar.com</title>
                <meta charset="utf-8">
        </head>
        <body>
                <?php
                echo "Mi primera página con PHP embebido en HTML - aprenderaprogramar.com";
                ?>
        </body>
</html>
```

Programa 1. Ejemplo etiquetas de delimitación en PHP.

La frase inicial **<!DOCTYPE html>** es una **declaración del tipo de documento**, en este caso se trata del **tipo html,** y debe definirse antes de comenzar con el apartado <html>. Se trata de una **indicación para el navegador web sobre que versión de html se ha empleado**. En este caso se indica que es la versión 4.01.

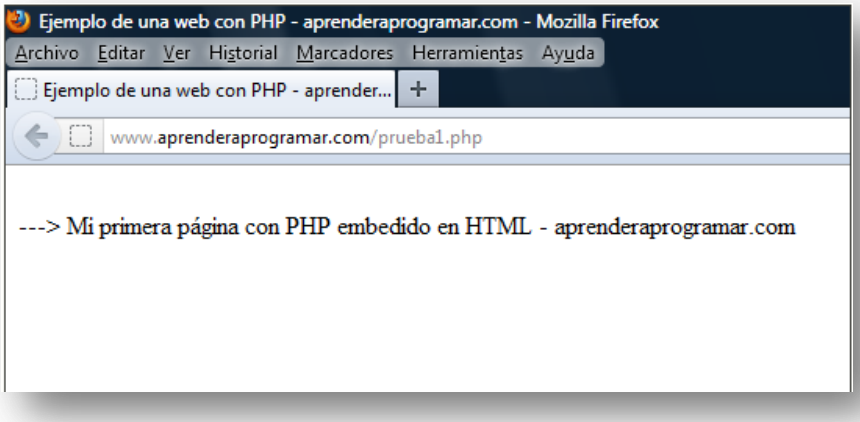


Figura 26. Respuesta web de programa 1

Para dotar al programa de **cierto orden** se emplean **etiquetas**. Estas se introducen **entre símbolos de mayor y menor**, y el **texto** o **comando** será introducido **entre la etiqueta de apertura** <*etiqueta*> **y cierre**</*etiqueta*>. Entre ellas destacan, además de las mencionadas HTML,HEAD y BODY:

- **Nobr**

La etiqueta <nobr> obliga al navegador a que muestre esa **frase sin saltos de línea**. El texto que queremos que no se parta debe ir dentro de las etiquetas de apertura (<nobr>) y de cierre (</nobr>).

- **Center**

La etiqueta nos permite **centrar párrafos**, **imágenes** o **tablas** dentro de nuestra Web.

- **Span**

La etiqueta nos permite **agrupar un conjunto de elementos** y así poder establecer unas reglas de estilo comunes para el conjunto.

Para realizar **saltos de línea** en la página web se inserta al final de la línea del comando la etiqueta **
.** Se puede sustituir esta orden por **nl2br**, introducida tras el comando echo para separar una frase en varias líneas:

```
$texto = “Uno\nDos\r\nTres”;

echo nl2br($texto);
```

El código anterior nos proporciona la siguiente salida:

```
Uno<br />

Dos<br />

Tres
```

```
<HTML>
<HEAD>
<TITLE>Ejemplo 10</TITLE>
</HEAD>
<BODY>

<H1>Tablas b&aacute;sicas</H1>

<TABLE BORDER="1">
<TR>
   <TH>Cabereca 1</TH>
   <TH>Cabereca 2</TH>
   <TH>Cabereca 3</TH>
</TR>
<TR>
   <TD>Dato 1</TD>
   <TD>Dato 2</TD>
   <TD>Dato 3</TD>
</TR>
<TR>
   <TD>Dato 4</TD>
   <TD>Dato 5</TD>
   <TD>Dato 6</TD>
</TR>
</TABLE>

</BODY>
</HTML>
```

Programa 2. Ejemplo de realización de una tabla en PHP.

También se podrán realizar **tablas** mediante las funciones **tr, td y th**. Cada **fila** de la tabla se indica mediante las etiquetas **<tr> y </tr>.** Las etiquetas **<th> y <td>** con sus correspondientes etiquetas de cierre, indican las **filas individuales dentro de cada fila**. **<th> y </th>** indican que se trata de **celdas** que sirven como **encabezado** de tabla y suelen visualizarse en negrita. Mientras que **<td> y </td>** indican que se trata de **celdas comunes**.

Estos comandos, junto con los de salto de línea se introducen entre los delimitadores *body,* que a su vez quedan insertados en los delimitadores de PHP, aportando a la página web creada cierto orden y estilo.

Para poder **interactuar con los clientes** se emplean **formularios HTML**. Estos formularios se **introducen** entre los símbolos **<FORM> y </FORM>.**

La marca <FORM> tiene que incorporar **dos parámetros**. Uno es constante, e indica **cómo se enviarán los datos**: **METHOD=POST** y otro indica la **página PHP que procesará la información** del formulario: **ACTION=pagina.php**. Es importante que la dirección del atributo ACTION sea relativa, porque si es absoluta (de la forma ACTION=http://www.dominio.com/pagina.php o ACTION=/directorio/pagina.php) sólo funcionará en un servidor (o una estructura de directorios) determinado.

Mediante la orden **INPUT TYPE** se crean **cajas de texto** en el programa:

```
<INPUT TYPE="texto" NAME="ciudad" VALUE=”pamplona” SIZE=8 MAXLENGTH=20>
```

Botones de selección: permiten **elegir uno** (y sólo uno) **de los elementos agrupados**. Es importante que todos los elementos agrupados tengan exactamente el mismo nombre (para que sean excluyentes) y distintos valor en VALUE (que será lo que identifique el seleccionado). Además, si se desea obligar al usuario a que seleccione uno de los elementos hay que poner el atributo **CHECKED** en alguno de ellos (pues en otro caso no aparecería ninguno seleccionado por defecto).

```
<BR><INPUT TYPE="radio" NAME="música" VALUE="1" checked>Flamenco
<BR><INPUT TYPE=" radio" NAME="música" VALUE="2">Pop
<BR><INPUT TYPE=" radio " NAME="música" VALUE="3">Rock
```

Cajas de selección: similares a los botones de selección, pero **se pueden seleccionar los elementos que desee** (uno, varios o ninguno). En este caso son independientes, por lo que casa uno tiene su nombre y el valor CHECKED si deseamos que por defecto aparezca marcado

```
<INPUT TYPE=”checkbox” NAME=”publi” VALUE=1> Marque si desea publicidad
```

Listas desplegables: permite **elegir entre uno o varios valores mostrados**. Entre la marca de apertura y la de cierre puede haber tantos elementos de selección como se desee. Existe el atributo SIZE que indica las opciones que se verán simultáneamente en pantalla y MULTIPLE que indica si es posible realizar una selección de más de un valor (con el atributo MULTIPLE su función es similar a la de las cajas de selección y sin él a la de los botones de selección).

```
<SELECT NAME=”provincia”>
<OPTION VALUE=”1” CHECKED>Sevilla
<OPTION VALUE=”2”>Huelva
</SELECT>
```

Estas con las bases para la creación de programas en PHP. Como ya se ha comentado anteriormente, la sintaxis de comandos empleados es similar a la de C, por lo que el conocimiento previo de este lenguaje puede facilitar la programación en PHP.

3.3. Línea de comandos

En Linux existen dos maneras principales de realizar una determinada tarea: a través de la interfaz gráfica o de la línea de comandos. La **línea de comandos** se conoce también como **consola o terminal.**

El terminal permite al usuario controlar el dispositivo mediante el **uso de comandos**. Estos comandos pueden unirse y combinarse para la creación de scripts más complejos que realicen tareas de manera eficiente.

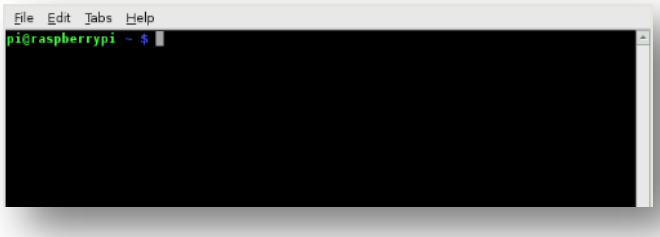


Figura 27. Terminal Raspbian.

En RaspBian, la aplicación de terminal empleada por defecto es **LXTerminal**. Es conocido como un **emulador de terminal** que imita los antiguos terminales de video en un entorno gráfico.

En el terminal aparecerán el nombre del usuario seguido del símbolo @ y del nombre del host.

Se van a **clasificar los comandos** según el tipo de función que realicen:

Comandos referentes a directorios:

- **cd** : este comando seguido del nombre de un directorio, permite navegar a través de la estructura de los directorios.

```
cd /home/pi/python_games
```

- **cd.** : permite ir al directorio anterior actual.
- **cd ~**: dirigirá al usuario al directorio inicial (/home/pi).
- **ls** : dará una lista de los ficheros y subdirectorios que contiene el presente directorio
- **pwd**: mostrará el nombre del directorio actual.
- **Mkdir**: crea un directorio con el nombre que le asignemos.

```
Mkdir directorio_nuevo
```

Comandos referentes a la ejecución, creación o lectura de archivos:

- **nano**: se trata de un editor de textos muy sencillo que viene preinstalado en la Raspberry.

  nano ejemplo.py : este comando permite crear o leer el archivo ejemplo.py.

- **python**: con este comando, seguido del nombre del archivo, permitirá ejecutar dicho programa de extensión .py.

```
python ejemplo.py
```

Comandos referentes a la instalación de programas:

- **apt-get**: se utiliza para manejar paquetes en distribuciones GNU/Linux basadas en Debian. Alternativamente, puedes utilizar el comando aptitude en vez de apt-get.

  - **apt-get install**: es la opción de apt-get que indica la instalación de uno o más paquetes.

    ```
    apt-get install nombre_del_paquete_de_instalación
    ```

  - **apt-get update**: opción de apt-get que sincroniza los archivos del índice de paquetes con los repositorios oficiales (dicho de otra forma, obtiene las actualizaciones).
  - **apt-get upgrade**: opción de apt-get que actualiza el sistema.
  - **apt-get remove**: este comando seguido del nombre del paquete permite desinstalar y eliminar cualquier paquete.

Algún comando hace cambios permanentes en el sistema. Dichos programas requieren de **privilegios *root*** (o administrador) para su instalación y configuración. El usuario *root* es el único usuario que tiene permisos para instalar paquetes en el sistema operativo.

sudo: da permiso temporal al usuario para poder ejecutar programas que requieren acceso a la raíz del sistema.

```
sudo apt-get install
sudo apt-get upgrade
sudo nano
…
```

O simplemente se podrá usar el comando sudo, tras el que se preguntará por la contraseña para obtener los permisos de administrador o raíz.

4. Implementación inicial

4.1. Equipos empleados

4.1.1. Rapberry Pi

Tras exponer los distintos modelos de Raspberry Pi existentes en el mercado, se ha concluido con la elección del **modelo 2B**.

La Raspberry Pi 2B es el **elemento principal** y sobre el que se ha basado el proyecto. Es importante conocer bien el alcance de dicho elemento, y de las diferentes opciones de programación que abarca.

La Raspberry Pi requiere de ciertos elementos externos para su correcto funcionamiento. Se pueden distinguir entre los **elementos indispensables** para el funcionamiento de la Raspberry, y los **elementos necesarios** para obtener las funciones características requeridas, en este caso el sistema de visualización de video en *streaming*.

Entre los elementos **indispensables** están la **alimentación** de la Raspberry y la tarjeta **microSD** con el S.O.

4.1.1.1. Alimentación

La Raspberry Pi emplea un **conector microUSB** para la alimentación. Puesto que se ha empleado el modelo 2B, requiere de **5V y 800 mA** para su correcto funcionamiento. Si la corriente de alimentación es menor a la necesaria, es posible que originen problemas.

Estos problemas aumentarán a medida que se añaden complementos a la Raspberry. Si la alimentación es insuficiente para la realización de ciertas funciones, la Raspberry se reiniciará. A su vez, si no se suministra suficiente corriente al iniciarse, esta dará errores y no concluirá su inicialización.

Por tanto, es importante conseguir un cargador que permita una correcta alimentación para evitar dichos problemas.

Para ello finalmente se ha empleado un cargador con salida de **5V y 2000mA**, que permite la correcta alimentación de la Raspberry y todos sus periféricos.

4.1.1.2. Tarjeta microSD

El **S.O. de la Raspberry** hay que **grabarlo** en una tarjeta **microSD**, que posteriormente se introduce en puerto correspondiente de la Raspberry para su lectura.

Se ha empleado una tarjeta microSD de **16Gb** con su adaptador para puertos SD de la marca SanDisk. Aunque una tarjeta de 4Gb es suficiente para la instalación de RaspBian, se ha elegido una tarjeta de mayor capacidad para no tener problemas con el almacenamiento, ya que se

pretenden instalar varios programas para poder realizar pruebas con ellos y concluir con la elección del más apropiado.

4.1.2. Conexión a internet

La **conexión a internet** de la Raspberry se ha realizado mediante un **cable Ethernet**. La Raspberry posee un puerto correspondiente, por lo que su conexión es muy simple. Existe un método alternativo para realizar la conexión a internet, mediante un adaptador WiFi que se inserta en el puerto USB.

La configuración de internet mediante **WiFi** se puede realizar de forma muy simple. Basta con entrar en el menú principal del interfaz gráfico, acceder a preferencias y a configuración WiFi. Se desplegará una ventana en la que se configurará el adaptador como wlan0, y en la opción de estado actual se pulsará en el botón escanear para obtener las redes WIFi disponibles. Tras encontrar la red requerida, se editará dicha red en el la opción gestión de red, y se añadirá la contraseña. Con esto quedará configurada la rd WiFi en este dispositivo.

Sin embargo, para este proyecto se ha empleado el primer método, por lo que simplemente se conectará el cable Ethernet al puerto correspondiente de la Raspberry.

4.1.3. Cámara RPi

Para cumplir con el objetivo final de este proyecto, es necesaria la utilización de una cámara que permita obtener imágenes y videos con la Raspberry.

Se ha empleado la **cámara oficial de la Raspberry**, la cámara RPi, que dispone de un **puerto adaptado** para su conexión directa. Esta cámara no requiere de configuraciones complejas como ocurre con las cámaras USB. Simplemente requiere de la **habilitación de la cámara** en la configuración principal de RaspBian.

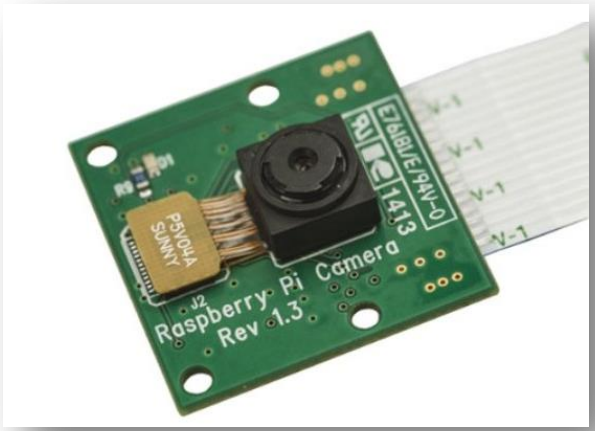


Figura 28. Módulo cámara RPi.

La cámara RPi que se ha empleado es la versión 1.3, con filtro IR, que posee una lente **de 5 megapíxeles** de la marca Sony. Tiene una **resolución de imagen de 2592 x 1994**, y **de video de 1080p**. En cuanto a la frecuencia máxima de video de 30 fps. **Soporta** Windows XP/Vista/7/8, **Linux** y Mac OS X.

4.1.4. Servomotores

Para **dotar de movimiento** a la cámara se han empleado **2 servomotores**, que van a realizar giros en dos ejes ortogonales, y con ello permitir que el rango de imagen de la cámara aumente considerablemente. Los servos se van a montar en una **plataforma** que permite este movimiento.

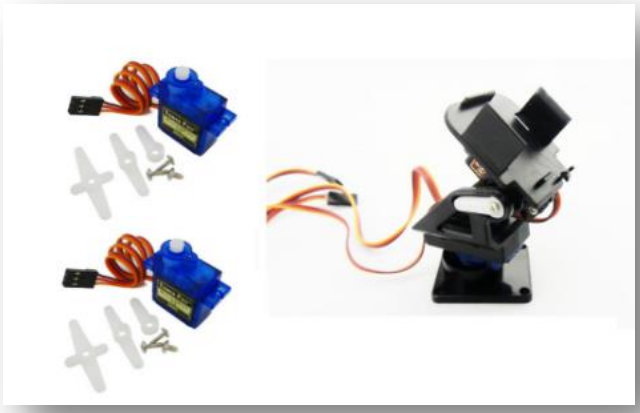


Figura 29. Servomotores Tower Pro más plataforma de montaje.

Los servomotores empleados son de la marca **Tower Pro**. Emplean una **alimentación de 5V**, una **conexión a tierra**, y una **señal de PWM**. Esta última señal es la que confiere de movimiento a los servos, y fija el giro y su posición.

Figura 30. Especificaciones del servo Tower Pro empleado

Para su alimentación se van a emplear los puertos de 5V el GPIO, ya que como se emplean pocos periféricos, la corriente que se le puede suministrar a los servos es suficiente para no dar errores. Sin embargo, si se quieren emplear más periféricos, ya sea desde los puertos GPIO, salida de video o de sonido, será necesaria la alimentación de los servos desde una fuente de alimentación externa.

4.1.5. Ratón y teclado.

Para poder **navegar por el interfaz gráfico** que ofrece RaspBian se van a utilizar un ratón y un teclado. Estos van conectados a los puertos USB, y se utilizarán para la **configuración del S.O. NOOBS**.

4.1.6. Monitor

Al igual que el ratón y el teclado, hace falta un **monitor** para poder emplear el **interfaz gráfico** de la Raspberry. Para ello hay que realizar la conexión entre un monitor y la Raspberry mediante una de las **salidas de video**.

Figura 31. Conector HDMI-VGA.

En este caso se ha empleado la **salida HDMI** y como monitor un televisión. Sin embargo, para realizar la conexión con un monitor de ordenador sin puerto HDMI se ha utilizado un adaptador de video HDMI a VGA.

4.1.7. Ordenador portátil

Para la realización del proyecto se ha requerido de un **ordenador portátil**. El ordenador portátil se ha utilizado para:

- **Búsqueda de información** referente al proyecto, tanto de la Raspberry como de los diferentes programas.
- **Descarga e instalación de los S.O.** en la tarjeta microSD.
- **Programación** a partir de Putty y VNC viewer.

Puesto que posee Windows 10, es compatible con Windows 10 IoT y se han podido realizar pruebas con cualquier S.O.

4.2. Programación

En este apartado se van a explicar los diferentes **programas probados** durante la realización del proyecto, comentando el funcionamiento principal de cada uno y las ventajas y desventajas que pueden ofrecer entre ellos.

Además de explicar el funcionamiento de dichos programas, se va a intentar a describir de manera general la programación principal de los mismos.

4.2.1. Cámara

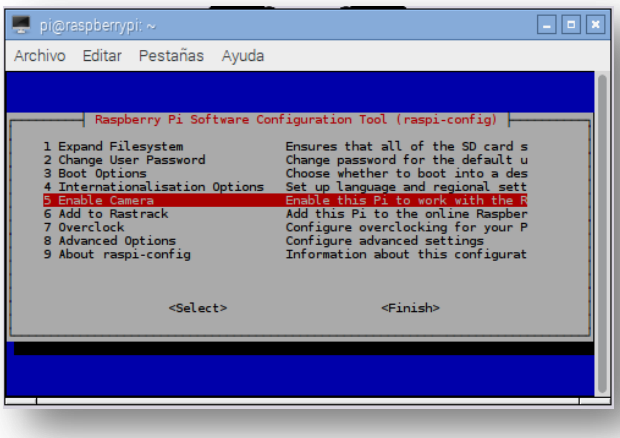
La búsqueda principal de este proyecto se ha realizado en el ámbito de la programación de la cámara, es decir, en la búsqueda de un programa que se adapte a los objetivos iniciales del proyecto.

Sin embargo, antes de experimentar con programas complejos, se ha tratado de obtener el correcto funcionamiento de la cámara a través de **programas** más **simples** creados a partir del **terminal o de Python**. Por ello, entre los programas explicados a continuación, se va a hacer referencia a dichos programas.

Lo primero que hay que realizar antes de comenzar con la programación, es **configurar la cámara** en RaspBian. Existen 2 maneras:

- A través de la **linea de comandos**:

Escribir en el terminal el comando sudo **raspi-config**.



Aparecerá un menú con varias opciones:

camera Enable/Disable -> seleccionar esta opción

Enable support for raspberry Pi camera? -> elegir YES

Would you like to reboot now? -> elegir YES

Figura 32. Menú raspi-config para habilitación de la cámara.

- Mediante el **interfaz gráfico**:

Para ello en el **menú principal** desplegable se seleccionará:

Preferences ->

Raspberry configuration ->

Cámara Enable/Disable -> Se habilita la cámara

Figura 33. Configuración de la Raspberry Pi mediante el interfaz gráfico.

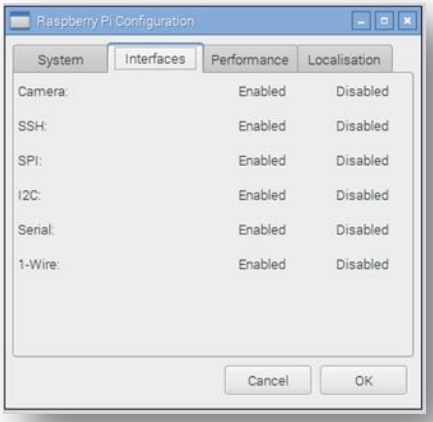


Figura 34. Habilitación de la cámara mediante el interfaz gráfico.

Se sale de la configuración, y se reinicia el sistema.

4.2.1.1. Comandos

La forma más sencilla de comenzar a usar la cámara de la Raspberry es mediante el **terminal**. Existen diferentes órdenes y comandos para realizar funciones con la cámara, ya sea sacar una imagen, realizar un video, ajustar los parámetros y calidad de la imagen, etc.

Se va a realizar a continuación una lista con los **comandos empleados**, y será una forma idónea de familiarizarse con este módulo.

- raspistill -o miprimerafoto.jpg

Este comando **tomará una imagen**, **de nombre miprimerafoto**, y la guardará en el directorio actual con la **extensión jpg**.

El **parámetro** –o permitirá obtener una salida, a la que asignaremos el nombre de la imagen. Sin embargo se pueden configurar muchos aspectos de una imagen:

-w, --width : configura el ancho de la imagen <medida>.

-h, --height : configura el alto de la imagen <medida>.

-q, --quality : configura la calidad del jpg <0 -100>.

-o, --output : nombre de la imagen de salida <nombre de la imagen>.

-t, --timeout : tiempo en ms que permanence la imagen en la pantalla tras sacar la imagen (si no se especifica es de 5s).

-d, --demo : Se inicia en modo demo.

-e, --encoding : format de la salida (jpg, bmp, gif, png).

-tl, --timelapse : modo timelapse. Obtiene imagines cada <t>ms.

-p, --preview : configura la imagen previa <'x,y,w,h'>.

-f, --fullscreen : pantalla previa completa.

-n, --nopreview : no emplea imagen previa.

-sh, --sharpness : configura la nitidez de la imagen (-100, 100).

-co, --contrast : configura el contraste (-100, 100).

-br, --brightness : configura el brillo (0, 100).

-sa, --saturation : configura la saturación (-100, 100).

-rot, --rotation : rota la imagen (90,180,270).

-hf, --hflip : giro horizontal.

-vf, --vflip : giro vertical.

-vs, --vstab : active el estabilizador de video

raspistill -o imagen_%d.jpg : Captura una imagen tras 5 segundos, donde %d será una secuencia numérica para no sobrescribir las imágenes.

raspistill -o imagen_%04d.jpg -tl 2000 -t 25000 :La opción -tl establece el tiempo entre fotos (en milisegundos) y la opción -t establece el tiempo total de la secuencia. En este ejemplo, se tomará una fotografía cada dos segundos (2000ms) durante un tiempo total de veinticinco segundos (25000ms). El %04d indica que las imágenes se guardarán con dígitos de 4 cifras (imagen_0001, etc.)

Ejemplo imagen: raspistill –o imagen.jpg –vf –t 100

Programa 3. Comando raspistill en el terminal

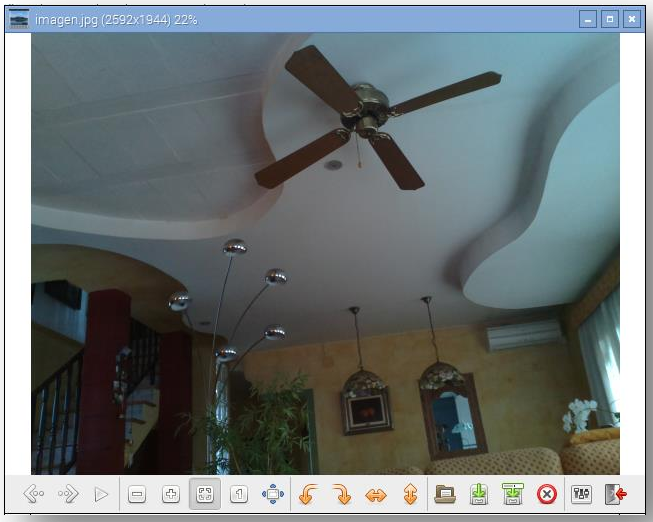
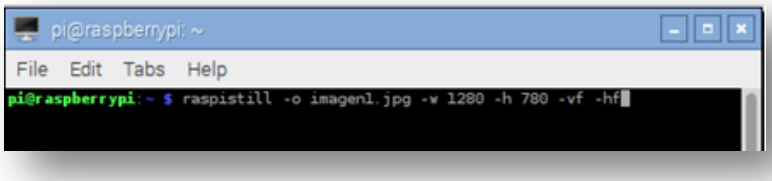


Figura 35. Imagen obtenida tras usar el programa 3.

Ejemplo imagen 1: raspistill –o imagen.jpg -w 1280 –h 780–vf –hf



Programa 4. Ejemplo de orden raspistill en el terminal.

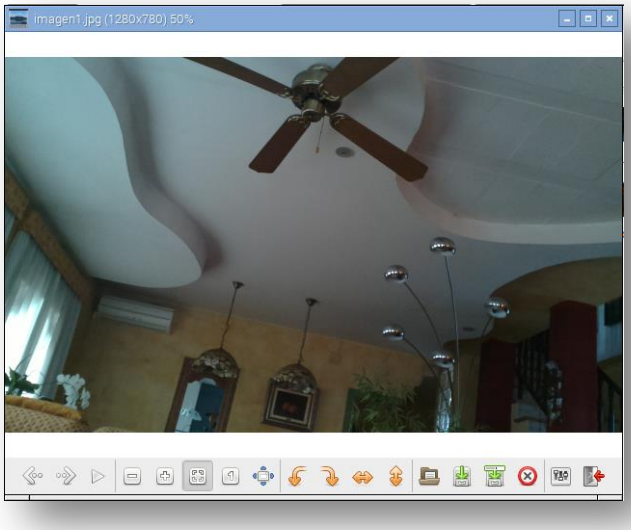


Figura 36. Resultado de ejecutar el programa 4.

Esta segunda imagen se ha volteado e incrementado la resolución respecto a la primera, de ahí que el formato de la imagen obtenida sea distinto.

- raspivid -o pruebavideo.h264 -t 10000

Mediante este comando se obtendrá un **video** en la pantalla de nuestro monitor de 10s de duración, cuyo nombre será prueba video. La **extensión** para la captura de videos es **h264**.

Los mismos parámetros comentados anteriormente para raspistill sirven para la configuración de raspivid. Sin embargo existen algunos comandos muy empleados para la grabación de video:

-**md**, --mode: elección entre los distintos modos de video.

Tabla 4. Función *mode* de raspivid.

Mode	Size	Aspect Ratio	Frame rates	FOV	Binning
0	automatic selection				
1	1920x1080	16:9	1-30fps	Partial	None
2	2592x1944	4:3	1-15fps	Full	None
3	2592x1944	4:3	0.1666-1fps	Full	None
4	1296x972	4:3	1-42fps	Full	2x2
5	1296x730	16:9	1-49fps	Full	2x2
6	640x480	4:3	42.1-60fps	Full	2x2 plus skip
7	640x480	4:3	60.1-90fps	Full	2x2 plus skip

-ifx, --imxfx: permite seleccionar distintos efectos para el video (también puede emplearse en imágenes).Estos son los efectos que pueden emplearse:

none, negative, solarise, sketch, denoise, emboss, oilpaint, hatch, saturation, cartoon, etc.

Para poder desarrollar **programas más complejos** que alcancen un propósito mayor, se pueden incluir estas funciones en **Python**.

4.2.1.2. Python

Para crear **programas** más complejos, que incluyan **mayor funcionalidad** y a su vez poder adaptarse mejor a los objetivos del proyecto, se ha realizado un estudio con las posibles opciones que presenta Python para la configuración de la cámara.

En primer lugar, hay que instalar la **librería** que emplea la **cámara** en Python. Esto se hace mediante el terminal, introduciendo el comando:

sudo apt-get install python-picamera

A continuación se abre desde el menú principal el **intérprete de Python IDLE 2**. Esto desplegará una nueva ventana en la que se podrá comenzar la programación.

En primer lugar hay que **definir las librerías** que se emplean en el programa. En este caso se van a emplear dos: **time y picamera**. Como se ha explicado en el apartado 3.1., estas librerías se instalan mediante el **comando import**:

```
Import time

Import picamera
```

Ejemplo Imagen 2:

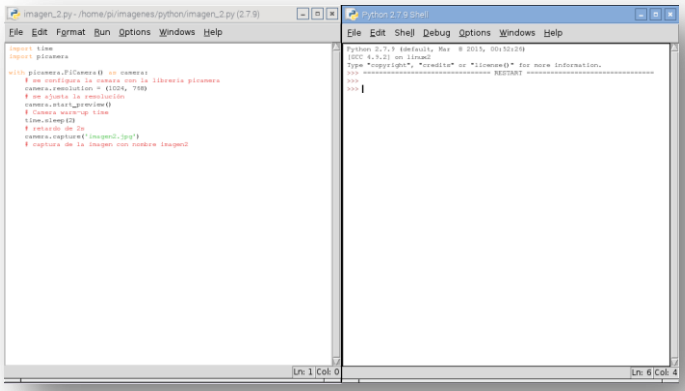


Figura 37. Intérprete IDLE tras el programa imagen2.py

imagen_2.py - /home/pi/imagenes/python/imagen_2.py (2.7.9)

File Edit Format Run Options Windows Help

```
import time
import picamera

with picamera.PiCamera() as camera:
    # se configura la camara con la librería picamera
    camera.resolution = (1024, 768)
    # se ajusta la resolución
    camera.start_preview()
    # Camera warm-up time
    time.sleep(2)
    # retardo de 2s
    camera.capture('imagen2.jpg')
    # captura de la imagen con nombre imagen2
```

Programa 5. Imagen2.py

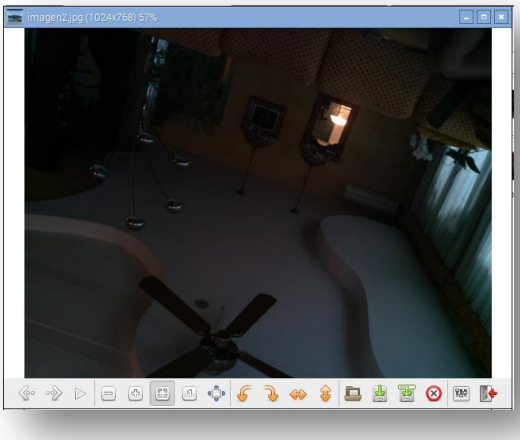


Figura 38. Imagen obtenida con el programa Imagen2.py

Ejemplo timelapse:

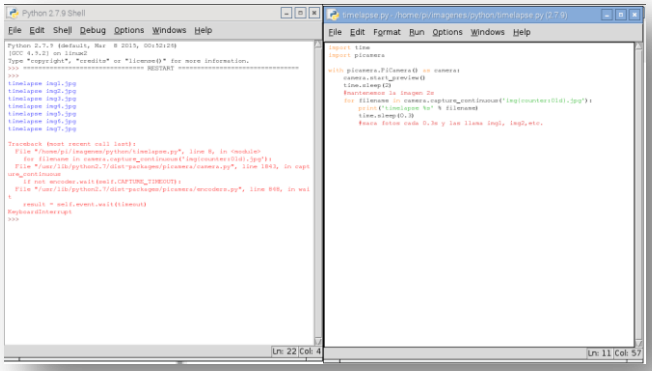
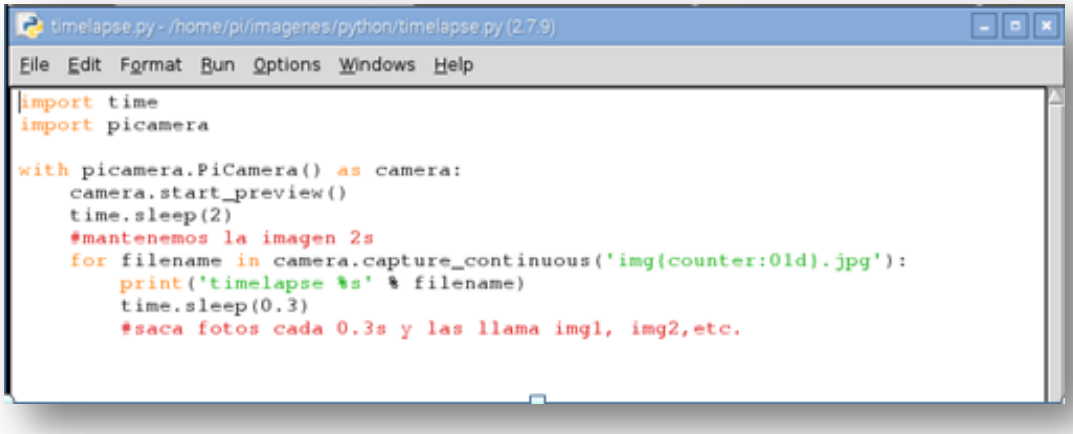


Figura 39. Intérprete IDLE tras el programa timelapse.py



Programa 6. timelapse.py

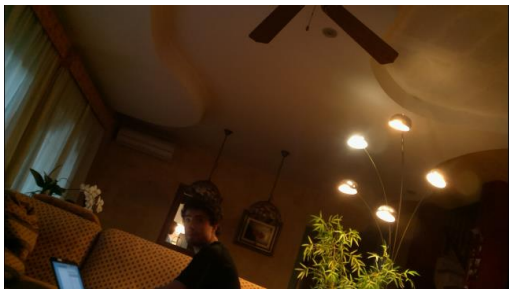


Figura 40. Secuencia timelapse, imagen 1.

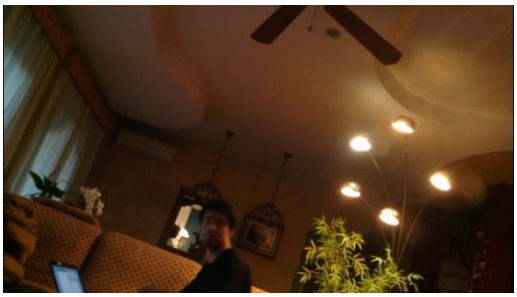


Figura 41. Secuencia timelapse, imagen 2.

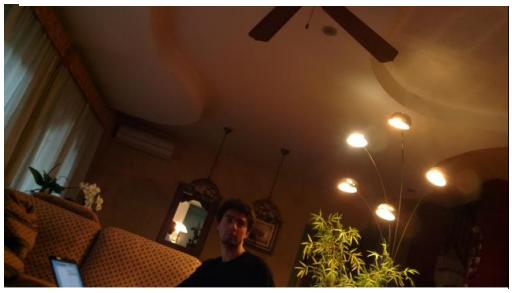


Figura 42. Secuencia timelapse, imagen 3.

Figura 43. Secuencia timelapse, imagen 4.

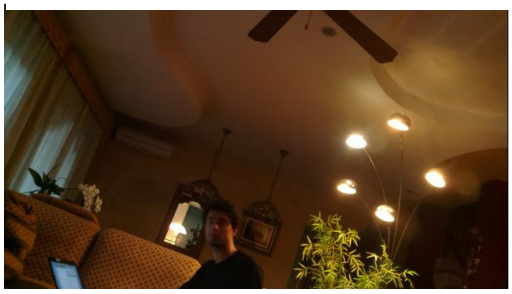


Figura 44. Secuencia timelapse, imagen 5.

Figura 45. Secuencia timelapse, imagen 6.

Ejemplo vídeo:

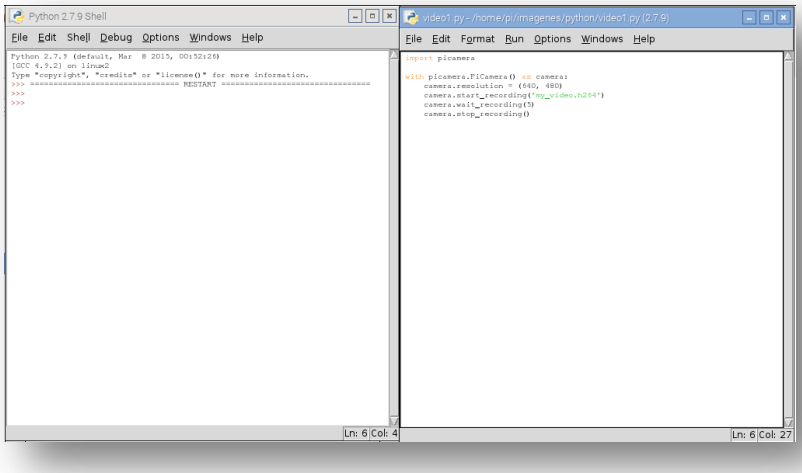


Figura 46.Intérprete IDLE tras el programa vídeo.py

video1.py - /home/pi/imagenes/python/video1.py (2.7.9)

File Edit Format Run Options Windows Help

```
import picamera

with picamera.PiCamera() as camera:
    camera.resolution = (640, 480)
    camera.start_recording('video.h264')
    camera.wait_recording(5)
    camera.stop_recording()
```

Programa 7. video.py

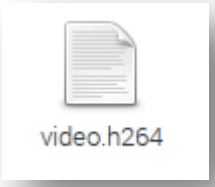


Figura 47. Archivo resultante del programa video.py

Estos ejemplos se han guardado dentro de la carpeta Imágenes en el directorio principal de la Raspberry.

Empleando las **librerías socket o netcat**, se podría conseguir **ver por la red local imágenes o videos**, sin necesidad de una compleja compilación de protocolos de red. Empleando a su vez la **librería Tkinter**, que permite crear una **interfaz gráfica**, se podría obtener un programa con la finalidad deseada.

Sin embargo, esto requiere conocimientos muy altos, ya que además de la programación de las librerías Tkinter y sockket o netcat, haría falta obtener conocimientos más avanzados sobre la captura y grabación de imágenes, sobre los protocolos de red, y sobre el direccionamiento y control de ficheros y directorios.

Un ejemplo de programación en el que se emplea el servidor socket, y se obtiene una imagen en el navegador es el siguiente:

video_internet2.py - /home/pi/imagenes/python/video_internet2.py (2.7.9)

File Edit Format Run Options Windows Help

```
import socket
import time
import picamera

# Conectar un cliente socket a 127.0.0.1:8000 (direccin local)
client_socket = socket.socket()
client_socket.connect(('127.0.0.1', 8000))

#Crear un objeto fichero fuera de la conexión
connection = client_socket.makefile('wb')
try:
    with picamera.PiCamera() as camera:
        camera.resolution = (640, 480)
        camera.start_preview()
        time.sleep(2)
        camera.start_recording(connection, format='h264')
        camera.wait_recording(20)
        camera.stop_recording()
finally:
    connection.close()
    client_socket.close()
```

Programa 8. Programa de empleo del servidor socket.

Sin embargo el resultado obtenido ha sido erróneo por problemas con la librería socket que viene preinstalada en Raspbian.

Como conclusión hay que destacar que la **programación en Python** en más **visual** que en la línea de comandos, ya que además de presentar un **sistema ordenado** mediante colores y sangrías, las órdenes para los ajustes de los parámetros emplean registros más coherentes y fáciles de memorizar.

4.2.1.3. Visor VLC

VLC es un reproductor de archivos, que se puede emplear a su vez para la **visualización de video a través de la red local**. Es en este ámbito en el que se quiere emplear el programa para acercarse a los objetivos del proyecto. **Asignando a la grabación una dirección IP y un puerto de salida**, el cual se utiliza en el programa VLC, se conseguirá fácilmente la visualización del vídeo.

Para emplear este visualizador, se requiere de la **instalación** del programa en el **dispositivo en el que se quiera ver**, además de en la **Raspberry**. Conectándonos a la red local mediante la dirección IP y el puerto de salida se conseguirá realizar la **descarga automática del video**. Sin embargo este video **no se podrá reproducir** hasta que finalice la grabación, y por tanto la descarga.

Con este último detalle este programa se aleja del objetivo principal, ya que no se consigue la reproducción en *streaming*, si no la obtención del video tras la grabación.

En primer lugar hará falta **instalar** el visualizador en la **Raspberry**. Para instalarlo se utilizará el terminal, en el que se introducirá la orden:

```
sudo apt-get install vlc
```

A continuación se utilizará a **función raspivid** con su correspondiente configuración, tras la que se escribirá la **función correspondiente al visor VLC**. Para ello se pondrá en el terminal:

```
raspivid -o - -t 0 -hf -w 800 -h 400 -fps 24 |cvlc -vvv stream:///dev/stdin --sout '#standard{access=http,mux=ts,dst=:8160}¬' :demux=h264
```

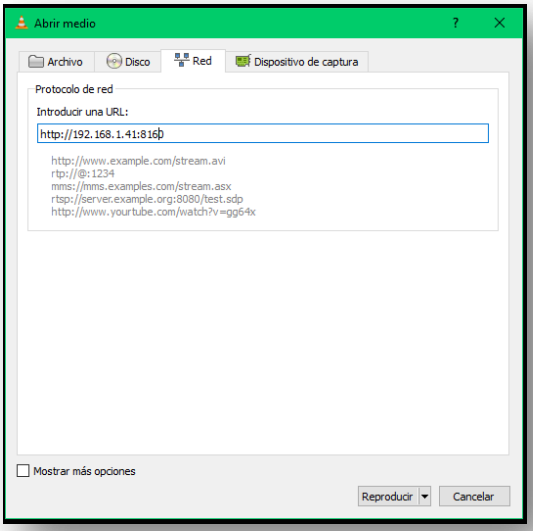


Figura 48. Menú volcado de red de VLC para el dispositivo remoto en que se realiza la descarga del vídeo.

Por último, en el dispositivo en el que se quiera descargar el video, con el **reproductor VLC** previamente instalado, se accederá al programa, y dentro de la opción "medio" se elegirá **abrir volcado de red**.

Una vez introducida la dirección IP y el puerto en el que se va a reproducir, los cuales se han definido en la función raspivid, comenzará la descarga del video.

4.2.1.4. Motion

Motion es un programa que **monitoriza la señal de vídeo** proveniente de una cámara. Es capaz de **detectar que** una parte significante de una **imagen ha cambiado**, es decir, que es sensible a **captar movimiento** entre las imágenes.

Motion es un programa de **visualización *streaming*** que provee al usuario de **múltiples opciones**. Estas opciones engloban la configuración de la cámara, en cuanto a la entrada empleada, si es mediante USB o video/TV card. A su vez se puede configurar la calidad de la imagen, si se desea que la salida sea en red local o a todos los usuarios, etc.

Todos los parámetros aparecerán en el **archivo motion.conf**, dentro del directorio de MOTION. Este directorio aparece tras la **instalación** de dicho programa, la cual se realiza mediante los siguientes pasos:

- En el terminal:

```
sudo apt-get install motion
```

Se instala el programa motion.

```
cd /tmp
```

Se accede al directorio tmp.

```
sudo apt-get install -y libjpeg62 libjpeg62-dev libavformat53 libavformat-dev libavcodec53 libavcodec-dev libavutil51 libavutil-dev libc6-dev zlib1g-dev libmysqlclient18 libmysqlclient-dev libpq5 libpq-dev wget https://www.scavix.com/files/motion-mmal.tar.gz
```

Motion aún no soporta el modelo cámara rpi, por lo que hace falta **instalar ciertas librerías.**

```
tar zxvf motion-mmal.tar.gz
```

Se descomprime el archivo descargado con las librerías.

```
sudo mv motion /usr/bin/motion
```

```
sudo mv motion-mmalcam.conf /etc/motion.conf
```

Se actualiza motion para las nuevas librarías.

```
sudo nano /etc/default/motion
```

Acceder mediante el editor de textos nano al archivo indicado y realizar el siguiente cambio:

start_motion_daemon=yes

```
sudo chmod 664 /etc/motion.conf
```

```
sudo chmod 755 /usr/bin/motion
```

```
sudo touch /tmp/motion.log
```

```
sudo chmod 775 /tmp/motion.log
```

Con esto se asegura que motion posee los permisos de usuario necesarios para ejecutarse tras el reinicio.

```
sudo nano /etc/motion.conf
```

Acceder al archivo con la configuración de motion a través del editor de textos nano y **configurar los siguientes parámetros**:

- daemon on : el demonio está siempre ejecutándose
- logfile /tmp/motion.log : almacenar el archivo de acceso en /tmp
- width 1280 : configurar la resolución del vídeo

  - height 720 : configurar la resolución del vídeo
  - framerate 2 : ajustar las imágenes por segundo del vídeo (si se quiere mejor calidad de video, subir el framerate, 30 como máximo para esta cámara)
  - pre_capture 2 : graba 2 imágenes antes de detectar movimiento.
  - post_capture 2 : graba 2 imágenes tras detectar movimiento en la imagen.
  - max_mpeg_time 600 : para que la secuencia de imágenes que forma el vídeo dure 10 minutos como máximo.
  - stream_localhost off : habilitar el acceso al vídeo en directo desde cualquier lugar, sino solo se podrá acceder al vídeo desde la Raspberry.
  - If you want to protect the live stream with a username and password, you should enable this:
  - stream_auth_method 2 : para proteger el vídeo con usuario y contraseña
  - stream_authentication USUARIO:CONTRASEÑA
  - Todos los parámetros están explicados con detalle en: motion config documentation.

- Por último hay que **reiniciar** el sistema

- Al navegar en la red local a la **dirección IP de la Raspberry**, al **puerto 8080** se obtendrá el video grabado con la cámara en tiempo real.

Motion es un sistema muy empleado para el uso de **cámaras de seguridad**, ya que se puede configurar para que **comience a grabar en cuanto note la presencia** de un objeto en el rango de alcance de la cámara.

A pesar de tratarse de un sistema con muchos adeptos, pues permite configurar el sistema para adaptarse a muchos requerimientos, al no tratarse este proyecto de obtener un programa de la videovigilancia, **no se ajusta totalmente a las necesidades de este proyecto**.

Además, se han encontrado varias objeciones más, puesto que los **archivos grabados son de un tamaño demasiado grande**, es difícil acceder a ellos a través del sistema y **no se puede iniciar la grabación cuando se quiera**, **ni tomar imágenes sueltas**. Por lo tanto se ha **desestimado** el empleo de **este programa**.

4.2.1.5. MJPG-STREAMER

Se trata de un programa diseñado para Raspberry cuya función principal es la **reproducción de vídeo en directo a través de internet**. Emplea comandos sencillos de la **librería picamera** para obtener imágenes, y tras obtenerlas en un directorio del sistema, manda dichas imágenes a internet.

Su empleo es un poco más complejo que los anteriores, ya que requiere de **introducir 3 comandos** en el terminal cada vez que se quiera emplear este programa. Sin embargo su funcionamiento posterior es muy simple, ya que se accederá mediante la **dirección IP y el puerto 8080** de la Raspberry a un menú inicial de MJPG-STREAMER, en el cual podremos navegar por distintas pestañas.

En cuanto a su **instalación**, se deberán seguir los siguientes pasos:

- En el terminal:

  sudo apt-get install libjpeg8-dev imagemagick libv4l-dev

  Instalar las 3 librerías que emplea el programa.

  sudo ln -s /usr/include/linux/videodev2.h /usr/include/linux/videodev.h

  Reemplazar el archive videodev.h erróneo.

  wget http://sourceforge.net/code-snapshots/svn/m/mj/mjpg-streamer/code/mjpg-streamer-code-182.zip

  Descargar el programa MJPG-STREAMER

  unzip mjpg-streamer-code-182.zip

  Descomprimir el archivo descargado anteriormente.

  cd mjpg-streamer-code-182/mjpg-streamer

  make mjpg_streamer input_file.so output_http.so

  Este comando permite ejecutar las órdenes indicadas mediante el interfaz gráfico.

  sudo cp mjpg_streamer /usr/local/bin

  sudo cp output_http.so input_file.so /usr/local/lib/

  sudo cp -R www /usr/local/www

  Instalar MJPG-STREAMER.

- **Reiniciar** el sistema

Para **emplear** el **programa** hay que compilar **3 comandos**:

- mkdir /tmp/stream

- raspistill --nopreview -w 1280 -h 800 -q 5 -o /tmp/stream/pic.jpg -tl 1 -t 9999999 -th 0:0:0 &
- LD_LIBRARY_PATH=/usr/local/lib mjpg_streamer -i "input_file.so -f /tmp/stream -n pic.jpg" -o "output_http.so -w /usr/local/www"

El primero de ellos **crea una carpeta en el directorio tmp** en la cual se guardará la imagen tomada a continuación. Esta carpeta se borrará cada vez que se reinicie la Rasspberry, por lo que habrá que introducir este comando cada vez que se encienda el sistema.

El segundo comando pertenece a la librería picamera vista anteriormente, cuya función es la de **obtener una imagen** con la cámara. Los **parámetros** que le siguen indican:

No se realizará una vista previa de la imagen.

Los ajustes de resolución y calidad (se han configurado para que la imagen sea de buena calidad).

La salida se guardará en la carpeta creada anteriormente con el nombre pic.jpg.

Se sacarán fotos cada milisegundo, y se mantendrá la imagen el máximo tiempo posible.

El parámetro th define los parámetros de las imágenes en miniatura (x:y:quality).

El último comando **inicia el programa MJPG-STREAMER**, en el cual se configura la imagen obtenida con el parámetro anterior como entrada, y la salida en la página local con puerto 8080.

Al acceder al puerto 8080 de la dirección IP de la Raspberry se accederá a la siguiente página:

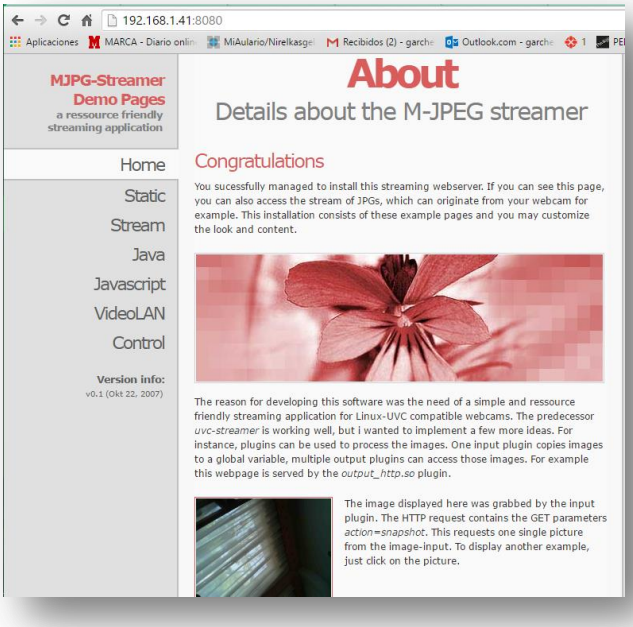


Figura 49. Página principal de MJPEG-Streamer.

Se trata del **menú principal de MJPEG-Streamer**, y en él se encuentra una pequeña descripción del programa. En la parte izquierda se pueden observar una serie de pestañas mediante las que accederemos a distintas páginas de este programa.

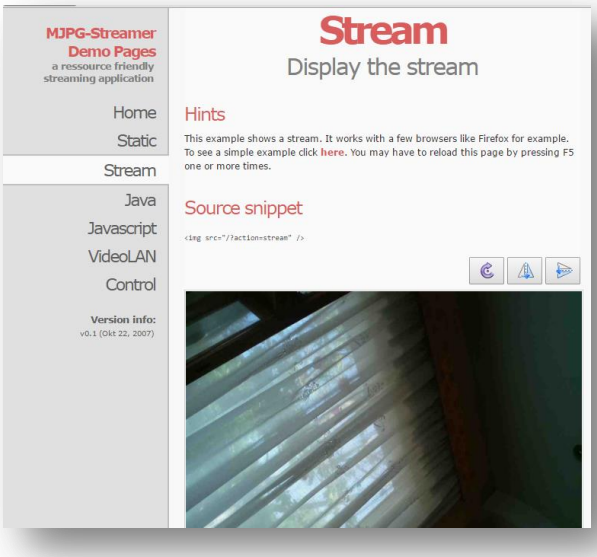


Figura 50. Pestaña Stream.

En la **pestaña Stream** se obtendrá el **video en directo**, de gran calidad y fluidez, y permitirá mediante 3 botones, girar la imagen y voltearla. Sin embargo esto se podrá realizar anteriormente mediante los parámetros de la librería picamera al introducir los comandos.

La **pestaña Static** mostrará una **imagen concreta del vídeo**, y la **ventana Video-LAN** indicará la **dirección web** en la cual se podrá obtener el **video** directamente.

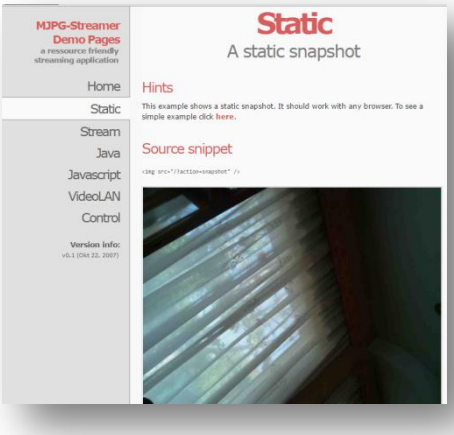


Figura 51. Pestaña VideoLAN

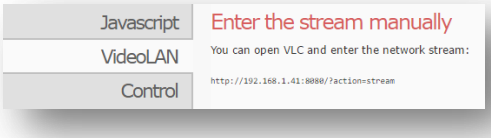


Figura 52. Pestaña Static.

Al igual que motion, este programa **no permite guardar las imágenes**, y la configuración de los parámetros de la imagen hay que configurarlos con rapistill, lo cual resulta más complejo. Por ello también se ha **desestimado** este programa para la obtención del proyecto.

4.2.1.6. RPI CAM WEB INTERFACE

RPi Cam Web Interface es un **interfaz de comunicación web para el módulo Pi Camera** de Raspberry, que puede ser abierto en el navegador (incluido en smartphones) y contiene las siguientes características:

- Se **puede iniciar, parar y reiniciar** el **vídeo** en directo de la cámara.
- Permite el **control de todos los parámetros** de la cámara.
- **Obtiene videos e imágenes** mediante el navegador, y los guarda en la memoria microSD de la Raspberry. Permite a su vez obtener dichas imágenes en el dispositivo del usuario.
- Tiene la opción de **video en timelapse**, es decir, en primer lugar obtener imágenes cada cierto retardo, y tras ello, convertir dichas imágenes en una secuencia de video.
- Además de la **descarga** de los **archivos**, permite **borrarlos** y **verlos mediante el navegador.**
- Captura imágenes al detectar movimiento a través del **módulo motion**.
- **Reinicia y apaga la Raspberry** a través del navegador

Hay que tener en cuenta que este programa sirve solamente para el módulo Pi Camera, no sirve para cámaras USB.

La **instalación** de este programa es muy sencilla. Bastará con seguir las siguientes instrucciones en el terminal para poder obtenerlo en la Raspberry:

```
git clone https://github.com/silvanmelchior/RPi_Cam_Web_Interface.git
```

Descargar el fichero RPI CAM WEB INTERFACE con los datos y configuración de los archivos y programas que emplea dicho programa.

```
cd RPi_Cam_Web_Interface
```

```
chmod u+x RPi_Cam_Web_Interface_Installer.sh
```

```
chmod u+x start.sh
```

```
chmod u+x stop.sh
```

Una vez dirigido al directorio del programa, se hacen ejecutables los archivos de instalación, inicio y parada. Con ello se podrá realizar dichas órdenes a través del interfaz gráfico.

```
./RPi_Cam_Web_Interface_Installer.sh install
```

Instala el programa anteriormente descargado.

```
./RPi_Cam_Web_Interface_Installer.sh autostart_yes
```

Configura el programa para que se autoinicie al arrancar la Raspberry.

En el **directorio de RPI WEB CAM INTERFACE** se podrán encontrar los **archivos ejecutables de inicio, parada e instalación** que se han configurado anteriormente. Estos archivos ejecutables irán seguidos de la terminación .sh:

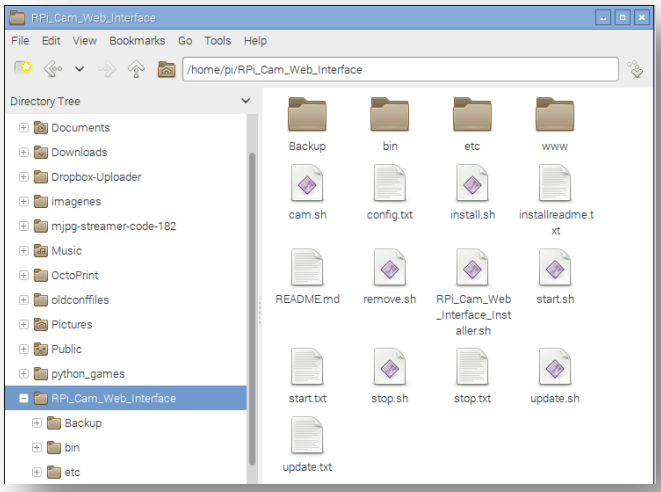


Figura 53. Directorio RPi_Web_Cam_Interface

Al instalar el programa, ya sea mediante la orden en el termina o pulsando el archivo *RPi_Web_Cam_Interface_Installer.sh* en el interfaz gráfico, aparece un **menú de configuración** en el que se pueden elegir las siguientes opciones:

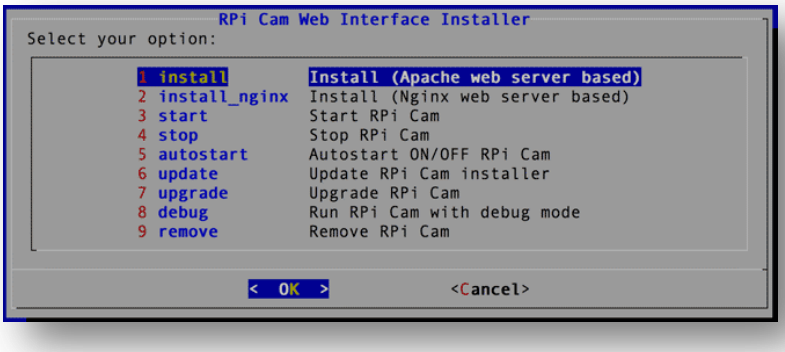


Figura 54. Menú de instalación de RPI WEB CAM INTERFACE

En primer lugar, se podrá elegir entre emplear Apache o Nginx como servidor web. En este caso se va a emplear **Apache**, por lo que se elegirá la opción 1.

Apache es un **servidor web** que viene preinstalado en la Raspberry y se puede emplear como **servidor de páginas web dinámicas** que utilizan **lenguaje PHP**.

A continuación, las 2 opciones siguientes permiten iniciar y parar el programa, mientras que la siguiente opción **configura si se quiere autoiniciar el programa** con la Raspberry.

También da la opción de actualizarlo o borrarlo.

En el **directorio media**, al que se accede a través de: var, www, html, se encuentran las **imágenes y vídeos realizados** mediante este programa.

Una vez instalado el programa, basta con ir al navegador, e introducir la **dirección IP de la Raspberry**, ya que esta es la **dirección que emplea Apache**.

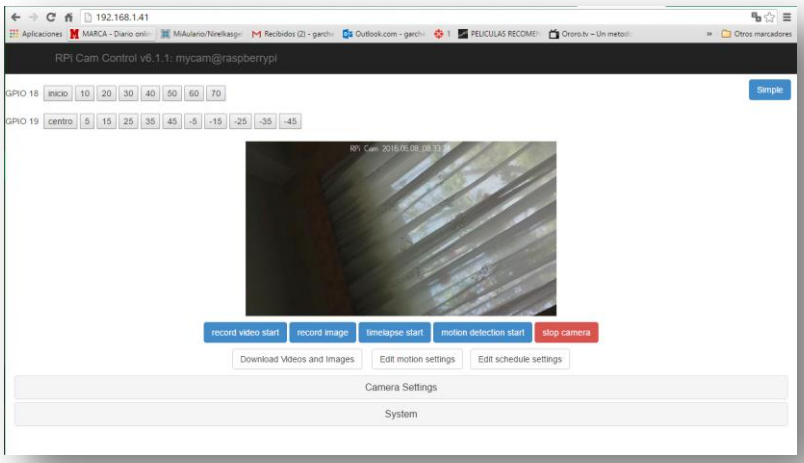


Figura 55. Página web principal de RPI WEB CAM INTERFACE.

El programa que aparece es el obtenido finalmente tras realizar ciertos cambios que se comentarán en el apartado 4.2.2.5.

Con esta última modificación se le han añadido las dos primeras líneas de botones, siendo el resto del programa el mismo. En él se pueden observar los **botones azules** para **realizar** una **imagen, video o timelapse**, al igual que la **detección de motion**, la cual hará falta activar. En **rojo** parecerá el **botón de apagar la cámara**.

Al pulsar este último, la imagen quedará congelada, y las funciones anteriores de capturas de imagen se desactivarán. El botón rojo ahora activará la cámara con la siguiente pulsación.

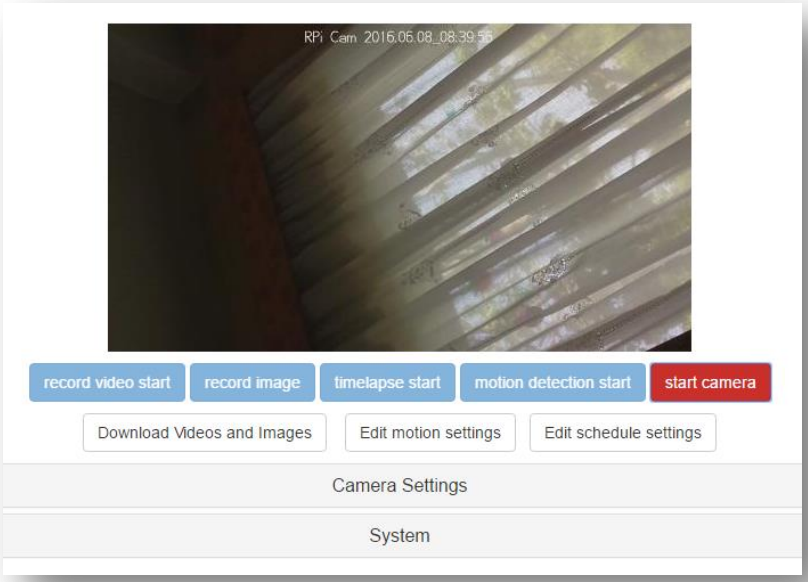


Figura 56. Imagen y botones del programa RPI WEB CAM INTERFACE.

El **botón de descarga de vídeos e imágenes** da acceso a una ventana en la que aparecerán los **archivos obtenidos y guardados en el directorio** media de la Raspberry.

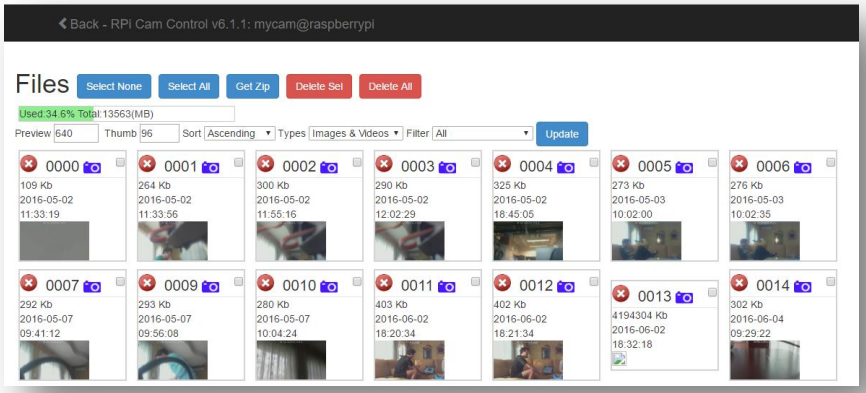


Figura 57. Página de descarga de vídeos e imágenes

Desde esta ventana se podrá realizar la **descarga** de los archivos, al igual que **borrarlos** o realizar la búsqueda de uno de ellos en concreto.

Volviendo a la página principal, también se tendrá acceso a **botones de configuración**, por un lado de la **cámara y sistema**, y por otro de **motion y de los registros**. Estos dos últimos no se van a emplear, ya que no se emplea motion, ni se requiere la configuración de unos registros diferentes.

En cuanto a las **opciones de la cámara**, se desplegará un menú con todos los **parámetros** de la cámara, con lo cual su modificación será muy sencilla. En este menú a su vez se podrá

configurar el nombre que aparece en la parte superior de la imagen en directo, la fecha y el tamaño y color de texto.

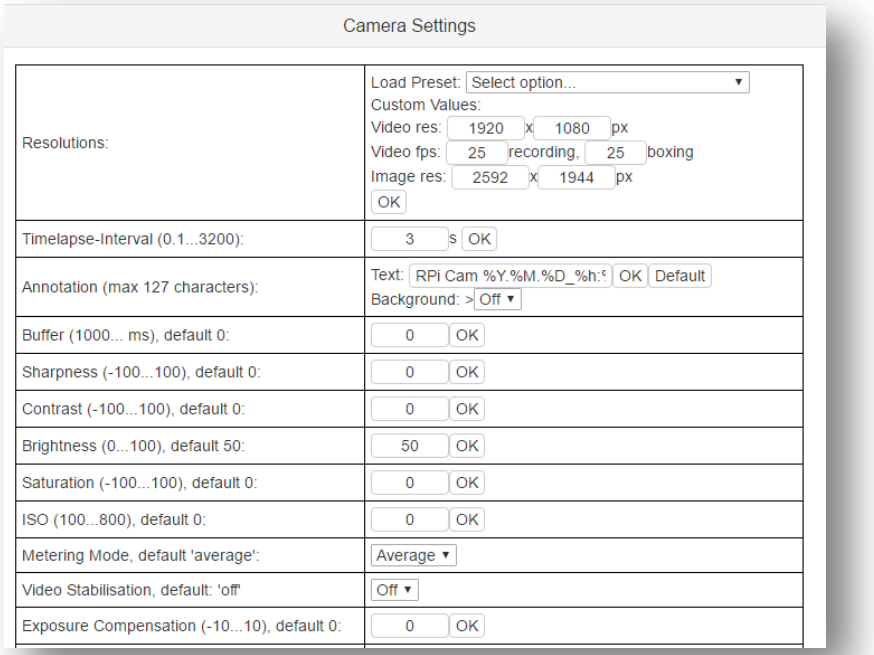


Figura 58. Tabla de los ajustes de la cámara que de despliega con el botón *Camera Settings*.

El cuanto a las **opciones del sistema**, se obtendrá un desplegable con las opciones de **reiniciar o apagar** el sistema, **restablecer la configuración inicial** y **cambiar el estilo** del programa.

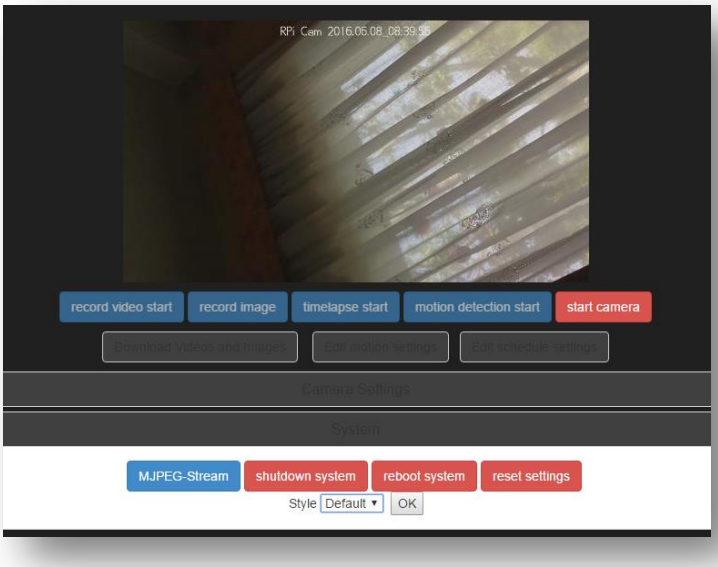


Figura 59. Cambio de estilo y botones del sistema.

Como se puede observar este **programa es muy completo**, y permite **alcanzar los objetivos** del proyecto en cuanto a la obtención de video en directo a través de internet, con la opción además de guardar imágenes y videos.

El modo de empleo es muy sencillo, y permite modificar gran variedad de características y parámetros, a la vez que se puede reiniciar o apagar el sistema a través del navegador.

Por todo ellos se ha elegido **RPI WEB CAM INTERFACE** como **programa principal del proyecto**. Sin embargo se realizarán posteriormente **modificaciones** para **introducir el funcionamiento de los servomotores** y conseguir **acceder** al programa **desde cualquier dispositivo**, sin necesidad de estar en la misma red local.

4.2.2. Servomotores

En este apartado se van a explicar las instrucciones básicas para el empleo de los servos en la Raspberry Pi. Sin embargo, en primer lugar se va a introducir el funcionamiento de los mismos para entender posteriormente los comandos empleados.

4.2.2.1. Funcionamiento

El **funcionamiento** de los servos se basa en la **modulación por ancho de pulso** (PWM) para **controlar el giro del motor**. Inicialmente se configura un **periodo base** para la señal PWM, y en función del **ciclo de trabajo** que se introduzca a dicha señal se controlará la posición del servo.

El **periodo** que se suele introducir a este tipo de motores es de **20ms**, lo que equivale a una **frecuencia de 50Hz**. A esta frecuencia se le conoce como **frecuencia de refresco**, es decir, se manda un pulso de control cada 20ms.

Como se ha podido observar en la figura 30 de las características del servo, los **pulsos de control** tienen un rango entre 0.5s y 2.5s. Esto quiere decir que para **0.5s** el servo se posicionará en la **posición izquierda** y para **2.5s** en la **derecha**, siendo **1.5s su posición central**. Aunque este rango de trabajo es relativo, ya que dependerá de periodo que se introduzca para la señal. Estos pulsos corresponden a periodos de 20ms.

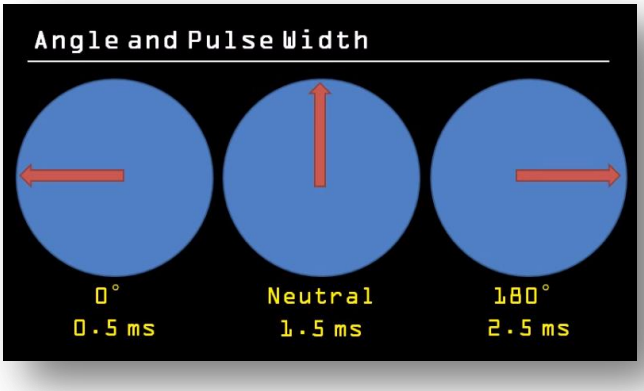


Figura 60. Ejemplo del giro de un servo en función del pulso.

Esto equivale **ciclos de trabajo** de **2.5%, 7.5% y 12.5%** para obtener aperturas de **0º, 90º y 180º** respectivamente.

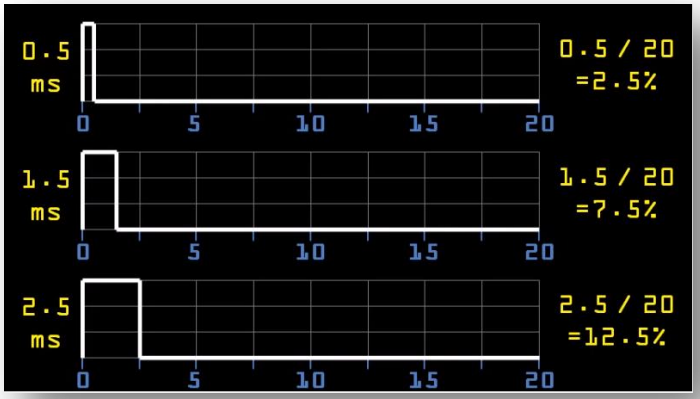


Figura 61. Ejemplo de la señal PWM para el movimiento de los servos.

Para **ciclos mayores al 12.5%**, se mantendrá en la **posición derecha**. Para este proyecto, el montaje de los servos en la plataforma obliga a modificar estos ciclos de trabajo para obtener los movimientos deseados.

En cuanto al **servo** que va a realizar el **giro**, situado en el puerto **GPIO 19**, se emplearán **ciclos de entre 1.5% y 3.5%**, siendo 2.5% la posición central. En principio no se deberían mandar pulsos menores a 2.5%, sin embargo no da ningún problema y se considera una adaptación idónea para las necesidades del proyecto. El giro establecido tendrá un rango un poco mayor a 90º.

Respecto al servo que va a fijar la **elevación** de la **cámara**, situado en el puerto **GPIO 18**, se van a emplear **ciclos de trabajo de entre 3.5% y 5%.** Corresponde a un **rango de 70º**, que representa el ángulo máximo de movimiento de la elevación de la plataforma.

Como se puede observar, los desplazamientos obtenidos no son los que se pensaban obtener teóricamente, sin embargo se ha realizado pruebas para ajustar el movimiento de los servos al requerido comprobando que los errores son mínimos.

4.2.2.2. Python

El método más sencillo para configurar el movimiento de un servo es mediante **Python**, ya que posee **librerías específicas** para el control de los mismos. Esta librería es **RPi.GPIO**, y para poder emplearla será necesario introducirla mediante la siguiente orden:

```
Import RPi.GPIO as GPIO
```

Esta librería **permite controlar los puertos GPIO**, y por tanto la **salida PWM** que contiene alguno de éstos. Por tanto el empleo de Python para los servos se simplifica mucho.

Para **configurar una salida** lo primero que hay que **definir** son los **puertos GPIO**. Estos se podrán configurar como **BOARD** (se refiere al pin por el número) o como **BCM** (se refiere a cada pin por su nombre).

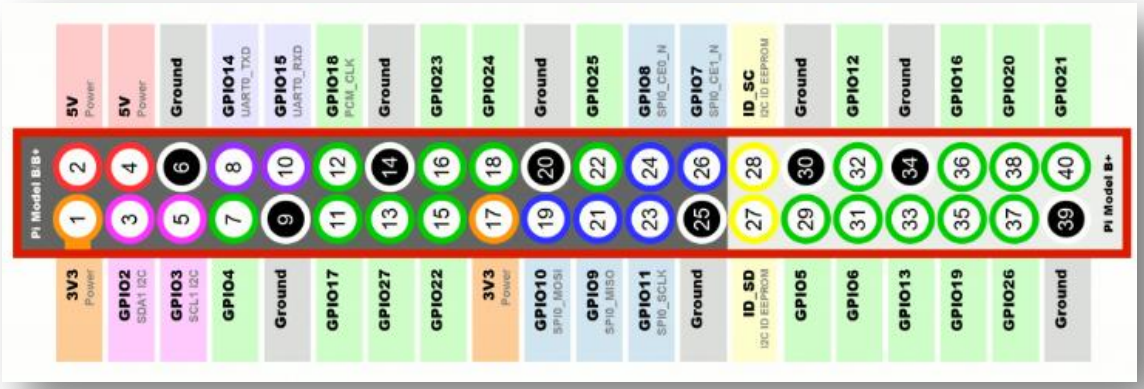


Figura 62. Puertos GPIO para el modo BOARD (números) y BCM (nombre).

En este caso se va a emplear el modo **BCM**, que se configura con la siguiente orden:

```
GPIO.setmode(GPIO.BCM)
```

Tras ello se podrá configurar la salida de los puertos GPIO 18 y 19 como **salidas PWM**:

```
GPIO.setup(18, GPIO.OUT)

pwm1 = GPIO.PWM(18,20)

GPIO.setup(19, GPIO.OUT)

pwm2 = GPIO.PWM(19,20)
```

El número 20 incluido en los paréntesis de la configuración del PWM los pulsos por segundo que se van a mandar, es decir, una **frecuencia de refresco** de 20Hz.

Para **utilizar** una **salida** como **PWM** hay que emplear **2 ódenes**:

```
pwm1.start (tiempo en ms del pulso)

pwm1.ChangeDutyCycle(tiempo en ms del pulso)
```

Con estas dos órdenes se **iniciará la salida PWM** a la que este referida la orden, y se **modificará el pulso de salida** para obtener movimiento en el servo.

Se va a realizar un **programa sencillo** de ejemplo para el **movimiento de los servos**. Este programa se encuentra en la carpeta servos, y se llama servo.py:

```
import RPi.GPIO as GPIO          #Importar la libreria RPi.GPIO
import time                      #Importar time para poder usar time.sleep

GPIO.setmode(GPIO.BMC)           #Poner la Raspberry en modo BOARD
GPIO.setup(19,GPIO.OUT)          #Poner el pin 21 como salida
p1 = GPIO.PWM(19,50)             #Poner el pin 21 en modo PWM y enviar 50 pulsos por segundo
p1.start(7.5)                    #Enviar un pulso del 7.5%  para centrar el servo
GPIO.setup(18,GPIO.OUT)          #Poner el pin 181 como salida
p2 = GPIO.PWM(18,50)             #Poner el pin 21 en modo PWM y enviar 50 pulsos por segundo
p2.start(7.5)                    #Enviar un pulso del 7.5% para centrar el servo
try:
   while True:                   #iniciar un bucle infinito

      p1.ChangeDutyCycle(4.5)    #Enviar un pulso del 4.5% : giro izquierda
      p2.ChangeDutyCycle(4.5)    #Enviar un pulso del 4.5% : giro izquierda
      time.sleep(0.5)            #pausa de medio segundo
      p1.ChangeDutyCycle(10.5)   #Enviar un pulso del 10.5%:giro derecha
      p2.ChangeDutyCycle(10.5)   #Enviar un pulso del 10.5%: giro derecha
      time.sleep(0.5)            #pausa de medio segundo
      p1.ChangeDutyCycle(7.5)    #Enviar un pulso del 7.5% para centrar el servo de nuevo
      p2.ChangeDutyCycle(7.5)    #Enviar un pulso del 7.5% para centrar el servo de nuevo
      time.sleep(0.5)            #pausa de medio segundo

except KeyboardInterrupt:        #Si el usuario pulsa CONTROL+C entonces...
   p1.stop()                     #Detenemos el servo
   p2.stop()                     #Detenemos el servo
   GPIO.cleanup()                #Se borra la configuración del puerto GPIO
```

A continuación se ha realizado un programa más complejo, en el cual **se podrá cambiar la posición de los servos mediante una aplicación de interfaz gráfico**. Para ello se va a emplear la **librería Tkinter**, que permite crear aplicaciones de interfaz gráfico para la interacción con el usuario. Este programa es **servotodo2.py**:

```
from Tkinter import *            #importar la librería Tkinter(clases, métodos y atributos)
import RPi.GPIO as GPIO          #importar la librería RPi.GPIO
import time                      #importar la librería time

GPIO.setmode(GPIO.BCM)           #Configuración de los puertos como PWM e iniciación de los mismos

GPIO.setup(18, GPIO.OUT)
pwm1 = GPIO.PWM(18, 20)
pwm1.start(3.5)
```

```
GPIO.setup(19, GPIO.OUT)
pwm2 = GPIO.PWM(19, 20)
pwm2.start(2.5)

class App:                          #Creación de una aplicación en Tkinter en la que se
                                    introducirán las funciones y métodos y atributos de
                                    Tkinter

  def __init__(self, master):       #Definición de un constructor en Tkinter mediante la
                                    función init con variable master. El parámetro self
                                    relaciona las distintas funciones entre sí.
    frame = Frame(master)           #Se crea un área principal de interfaz gráfica de Tkinter
                                    en la que aparecen 2 barras de interacción horizontales
                                    de valores 0-75 (línea 1) y -45-45 (línea 2), en las que se
                                    relacionará la salida de estas con las funciones
                                    posteriores.
    frame.pack()
    scale1 = Scale(frame, from_=0, to=75 ,
        orient=HORIZONTAL, command=self.update1)
    scale1.grid(row=0 )
    scale2 = Scale(frame, from_=-45, to=45,
        orient=HORIZONTAL, command=self.update2)
    scale2.grid(row=1)

  def update1(self, angle1):        #Creación de una función que actualiza el ciclo de
                                    trabajo del puerto 18 en función del valor obtenido de
                                    la primera barra de interacción anterior
    duty1 = float(angle1) / 56 + 3.5
    pwm1.ChangeDutyCycle(duty1)

  def update2(self, angle2):        #Creación de una función que actualiza el ciclo de
                                    trabajo del puerto 19 en función del valor obtenido de
                                    la segunda barra de interacción anterior
    duty2 = float(angle2) / 45 + 2.5
    pwm2.ChangeDutyCycle(duty2)


root = Tk()                         #Se crea una clase Tkinter.Tk, llamada ventana root,
                                    que es la que aparecerá en el monitor. Se le podrá
                                    llamar como se quiera.
root.wm_title('Elevar/Girar')       #Se le introduce nombre a la ventana root.
app = App(root)                     #Correspondencia entre la aplicación creada en Tkinter
                                    y la ventana root.
root.geometry("300x100+0+0")        #Dimensiones de la ventana root.
root.mainloop()                     #Hace que se mantenga la ventana root visible.
```

Al ejecutar este programa aparecerá la ventana Elevar/Girar creada con las barras de interacción para el movimiento de los servos.

Figura 63. Ejemplo 1 de la interfaz gráfica del programa servotodo2.py

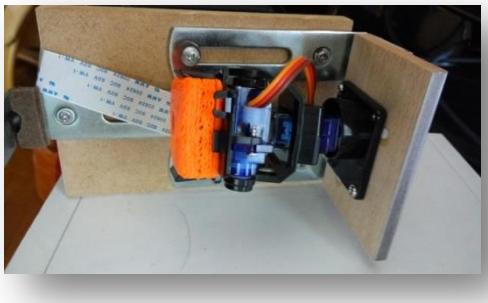


Figura 64. Ejemplo 1 del movimiento de los servos del programa servotodo2.py

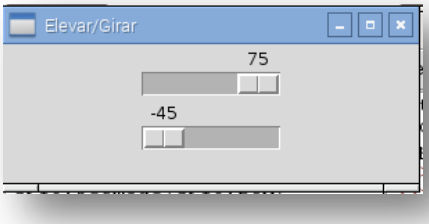


Figura 65. Ejemplo 2 de la interfaz gráfica del programa servotodo2.py

Figura 66. Ejemplo 2 del movimiento de los servos del programa servotodo2.py

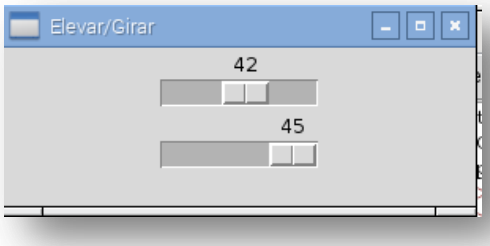


Figura 67. Ejemplo 3 de la interfaz gráfica del programa servotodo2.py

Figura 68. Ejemplo 3 del movimiento de los servos del programa servotodo2.py

A su vez se han creado programas más sencillos que realizan el movimiento de uno de los dos servomotores en lugar de los dos unidos, y se podrán encontrar en el mismo directorio que estos programas.

Sin embargo con estos programas no se va a conseguir la unión entre el vídeo en streaming y el movimiento de los servos. Esta unión, como se explica más adelante se va a realizar mediante la ejecución en PHP de programas Python.

Para ello se van a crear programas más sencillos, que simplemente configuran el ángulo deseado para uno de los motores. Esto se hace ya que va a ser imposible acceder al interfaz gráfico creado por Tkinter a través del navegador. Por ello, y puesto que se ha aprendido a crear botones en lenguaje PHP, se va a realizar un botón para cada configuración deseada.

Se ha optado por esta solución tras ver la imposibilidad de exportar un valor del programa en PHP e intentar introducirlo en el programa Python. Con esto se habría conseguido realizar el movimiento de los motores empleando un solo programa Python y un cuadro de texto en PHP en el que introducir el valor de ángulo deseado.

4.2.3. Dropbox

Dropbox es un famoso **servicio de almacenamiento de archivos en la nube**, que ha querido también ofrecer su producto en la plataforma Raspberry Pi. La misión de su programa es ofrecer un servicio para almacenar y sincronizar archivos en línea, pudiendo compartir dichos archivos entre usuarios y dispositivos.

Se ha querido emplear este programa para poder **obtener las imágenes y videos de manera sencilla en un dispositivo externo a la Raspberry**. Una vez configurado este programa, la subida de archivos a la red se realizará a través de un comando, de ahí la gran simplicidad de empleo de este programa.

Para poder emplear este programa, hace falta tener una **cuenta en Dropbox**. Además de tener una cuenta, es necesario crear una **aplicación en Dropbox**, ya que se pedirán los datos de dicha aplicación para poder emplear Dropbox en la Raspberry.

Una vez obtenida esa cuenta la instalación se realizará por medio del terminal:

```
git clone https://github.com/andreafabrizi/Dropbox-Uploader.git
```

Se descarga Dropbox-uploader

```
cd Dropbox-Uploader
```

```
./dropbox_uploader.sh
```

Se ejecuta el archivo anterior de la carpeta Dropbox-Uploader.

En este momento se pedirán el nombre de la aplicación y si se desea que los archivos del sistema se suban a la carpeta de la aplicación o a Dropbox por completo.

Tras esto, se pedirán la clave y contraseña para esta aplicación, las cuales se obtienen al crear la aplicación. Con esto quedará instalado Dropbox en la Raspberry.

En cuanto a su **uso**, se podrá elegir si emplear el **terminal o Python**, puesto que existen ordenes sencillas en los dos casos.

En **terminal**:

```
cd home/pi/Dropbox-Uploader
./dropbox_uploader.sh upload /home/pi/nombre_del_archivo  nombre_con_el_que
_se_quiere_subir
```

En **Python**:

```
from subprocess import call
photofile = "/home/pi/Dropbox-Uploader/dropbox_uploader.sh upload
/home/pi/imagen01.jpg mi_primera_foto.jpg
call ([photofile], shell=True)
```

No solo **permite subir** (*upload*) **archivos** a la nube, sino que también permite realizar más funciones, como **descargar** un **archivo** (*download*), **borrarlo** (*delete*), **moverlo de** ubicación (*move*), **realizar una lista** (*list*) o **compartirlos** (*share*).

El objetivo final de este programa es obtener los archivos requeridos en cualquier dispositivo, independientemente de estar en la red local o no. Sin embargo, requiere del empleo de ciertos comandos para que funcione, por lo que **hace falta acceder a la Raspberry para poder posteriormente obtener los archivos de forma remota**.

Por tanto, **este programa puede resultar muy interesante si se introduce en un programa más complejo**, en el que a la vez de sacar la foto, la suba directamente a la nube.

4.2.4. Exportar una red local

Hasta ahora se ha comentado que para poder ver o emplear los programas se requería de **acceso a la red local**, tanto para los programas de visualización de video como para los SSH o VNC viewer. Por tanto el empleo de la Raspberry quedaba reducido a la red local en la que estuviese conectada.

Para obtener un programa de visualización en directo de manera remota, se ha querido aumentar el rango de visualización, y para ello se ha intentado **exportar una dirección local a la red de internet.**

Se han encontrado **dos maneras** de realizar esto, una mediante **direcciones dinámicas y la configuración del router**, y la segunda mediante **programas existentes** en internet con acceso a Raspberry.

4.2.4.1. Direcciones dinámicas

La forma más empleada para exportar una dirección local a la red es mediante direcciones dinámicas. En primer lugar hay que entender de qué se trata este concepto.

La IP fija (estática) y la IP dinámica:

> Cuando la IP es **fija** esta combinación numérica **no varía**, siempre es la misma y cuando es **dinámica**, si **varía**, por tanto, puede ser que cada vez que conectemos, apagamos y encendemos el *router* o bajo determinadas circunstancias particulares se asigne una diferente.
>
> Al contratar la conexión a internet, lo normal es que contemos con direcciones IP dinámicas, si se quisiera estática o fija habría que pagar por ello.
>
> Por tanto, **dentro de la propia red local se tendrán direcciones fijas,** con las que se podrá acceder a los distintos aparatos que estén conectados a ella. Sin **embargo al router** o red general **se le asigna una dirección IP distinta cada vez que se conecta a la red.** Esto imposibilita acceder a la dirección local a través de internet, sin emplear la red local.

Sin embargo hay **programas** que **permiten asignar a una dirección de red local una dirección fija en la red global**. Para ello se **asigna** a la **dirección dinámica entrante al *router***, y perteneciente a la dirección de red local elegida, **una dirección IP constante**.

Para emplear este tipo de programas hace falta entrar en la **configuración del *router***, puesto que hay que **permitir el acceso a la dirección web de la Raspberry** Esto se hace para que el programa en cuestión sea capaz de detectar dicha dirección dinámica y le establezca una dirección web fija.

El *router* incluye un firewall o cortafuegos que a la vez que protege a la Raspberry Pi de Internet, imposibilita acceder a ella de manera remota. El acceso a la dirección web de la Raspberry se basa en la **deshabilitación** de este **firewall** para la Raspberry.

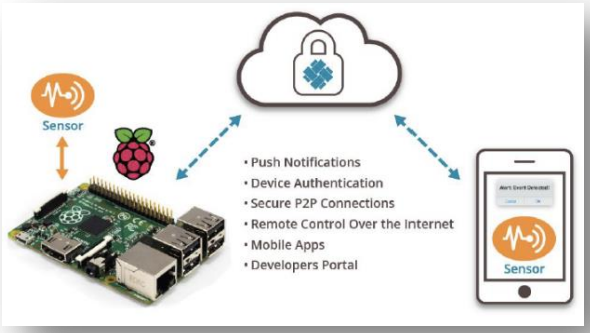
Un programa de este tipo es **No-IP**, el cual **genera** una **dirección** web a la que debemos asociar la dirección IP de la Raspberry Pi. Al asociar esta dirección conseguiremos que el **router redireccione la IP de la Raspberry a la de la página web creada**, haciendo así accesible al exterior la página deseada.

Incluso con este programa se podría acceder al SSH, a un visor de interfaz gráfico como VNC viewer o a cualquier puerto de salida de red de la Raspberry.

El **problema** de esta solución es que en la red en la que se va a emplear la Raspberry, no se puede entrar a la **configuración de la red local**, por lo que esta opción no se ha podido emplear.

4.2.4.2. Weaved

El **kit de Internet de las Cosas (IoT) de Weaved Inc**., una startup de Silicon Valley, proporciona una manera libre, fácil de usar y de **acceso** eficiente a la Raspberry Pi desde cualquier lugar en **Internet**. Sólo se necesita que **instalar el kit IoT de Weaved** para **acceder a través** del **portal web de Weaved** o la aplicación móvil.



F

igura 69. Conexión Weaved-usuario

Para su **instalación** hay que realizar los siguientes pasos:

1.- Comprobar que el **servicio** (WebIOPi, Apache, SSH…) al que se desea acceder está **corriendo** en la Raspberry.

- SSH en el puerto 22
- Web (http) en el puerto 80
- VNC en el puerto 5901

2.- **Crea** una **cuenta** gratuita en: https://developer.weaved.com/portal/

3.- **Descarga** el **instalador del Kit IoT de Weaved** en la Raspberry utilizando una ventana terminal o conexión SSH:

```
sudo apt-get install weavedconnect
```

4.- **Ejecuta** el **instalador**:

```
sudo weavedinstaller
```

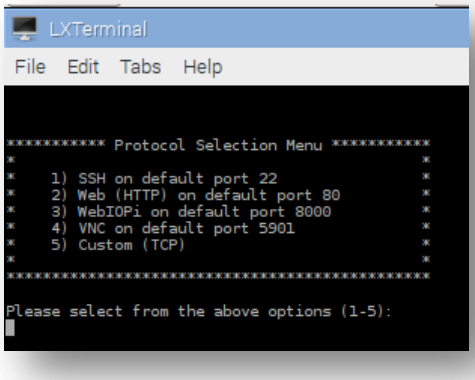
5.- **Inicia sesión** en el instalador utilizando la cuenta creada en el paso 2.

Pedirá el nombre de usuario y contraseña

6.- **Eligir el servicio** al que se desea acceder, puede ser Apache, SSH, WebIOPi o un servicio personalizado.

Figura 70. Elección de servicio Weaved.

Si se **selecciona 1**, se obtendrá acceso al **SSH**, y aparecerá el mensaje de la Figura 71:



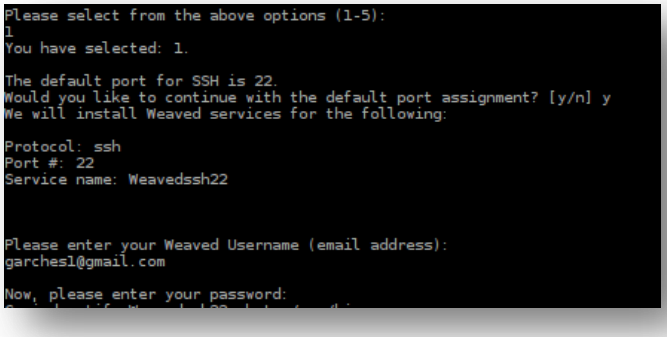
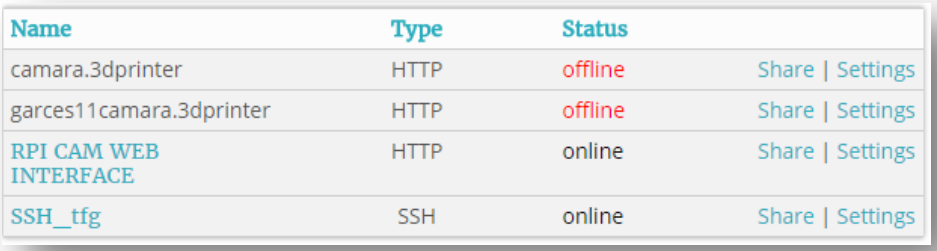


Figura 71. Servicio SSH Weaved.

Una vez introducido el usuario y contraseña, aparecerá un mensaje de que la instalación ha sido correcta y pedirá el **nombre para este servicio**. En la **cuenta Weaved** aparecerá el enlace a este nuevo servicio.

Siguiendo estos pasos se podrá configurar **Weaved** para el **servicio** requerido, en este caso para **Apache** y el **puerto 80**, que será el que emplee el programa principal del proyecto.



Name	Type	Status	
camara.3dprinter	HTTP	offline	Share | Settings
garces11camara.3dprinter	HTTP	offline	Share | Settings
RPI CAM WEB INTERFACE	HTTP	online	Share | Settings
SSH_tfg	SSH	online	Share | Settings

Figura 72. Vínculos de Weaved para las direcciónes fijas que asigna a las direcciones dinámicas de la Raspberry

El enlace a RPI WEB CAM INTERFACE llevará directamente a la página web correspondiente, mientras que si se accede mediante en el servicio SSH aparecerá la siguiente ventana:

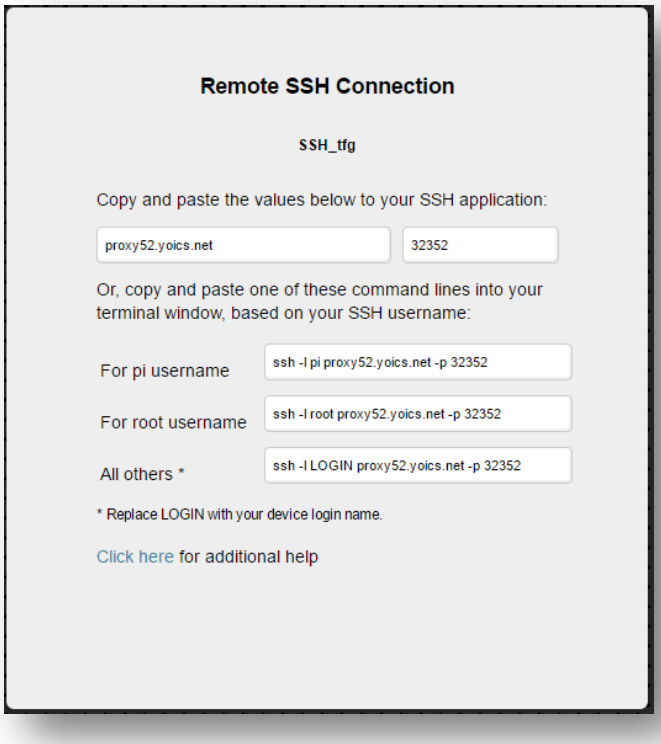


Figura 73. Enlaces para el empleo de Weaved como SSH.

Se deberán sustituir los dos primeros valores en las casillas de Host name y Port del SSH empleado para acceder al terminal de la Raspberry.

La información de Weaved respecto al *alias*, *Device* UID y *Device secret* pedidas durante la instalación se guardan en el archivo de licencia:

/etc/weaved/services/Weaved web80.conf

Este servicio se iniciará junto con la Raspberry, si se desea detenerlo, iniciarlo o reiniciarlo hará falta introducir el siguiente comando:

```
sudo /usr/bin/Weavedweb80.sh start|stop|restart
```

Como se ha visto en la explicación anterior, Weaved es una forma más sencilla de obtener una dirección web que mediante las direcciones dinámicas y la configuración del router. Por tanto **se ha empleado este servicio para conseguir exportar a internet la dirección local correspondiente al programa RPI WEB CAM INTERFACE**.

Se ha podido observar a su vez la **posibilidad** de **exportar el interfaz gráfico mediante el puerto 5091**. Esto posibilita la opción de controlar los servos mediante el interfaz gráfico, a la vez que con Tkinter y la librería picamera se podría obtener una ventana en el interfaz gráfico con la imagen en directo. Es decir, se podría realizar en Python un programa simple en el que se obtendría una imagen y el control de los servos. Sin embargo este programa tendría limitaciones, como el obtener imágenes o videos, la configuración de los parámetros, etc.

Para cumplir el objetivo final del proyecto haría falta realizar un programa que realice timelapse ajustando sus parámetros mediante la función raspistill. Por tanto, el programa RPI WEB CAM INTERFACE es más completo y adaptable a las necesidades del usuario, por tanto se va a mantener la decisión de emplear dicho programa en lugar de crear uno desde el inicio.

4.2.5. Configuración de la Raspberry

Una vez explicado el funcionamiento de la Raspberry, de los S.O. y de los programas que se van a emplear, se va a realizar una descripción de los **pasos seguidos para la obtención e instalación de los S.O. y los programas empleados**.

Se van a explicar los **pasos realizados** para la configuración de la Raspberry:

- Descarga e instalación del S.O.
- Configuración de Raspbian
- Configuración de los periféricos
- Instalación de los programas
- Implementación de RPI WEB CAM INTERFACE

4.2.5.1. Descarga e instalación del S.O

El **S.O.** elegido para este proyecto ha sido **Raspbian**, sin embargo no se va a instalar directamente, sino que se va a **emplear NOOBS** para ello, ya que simplifica mucho la instalación del S.O. elegido.

En primer lugar se va a proceder a la **descarga de esta aplicación**, la cual se puede obtener en la página oficial de Raspberry, www.raspberrypi.org, en el apartado de descargas.

Una vez descargado el sistema se debe **cargar en la tarjeta microSD**. Para ello hay que **borrar el contenido de la SD** con la aplicación **SDFormatter** y expandir la capacidad de la tarjeta al

máximo. A continuación se **grabará** el archivo descargado en la microSD empleado la herramienta de instalación **WIN 32 Disk Imager**.

Con esto la **aplicación NOOBS** quedará **instalada en la tarjeta de memoria**, y contendrá los datos necesarios para la instalación de Raspbian.

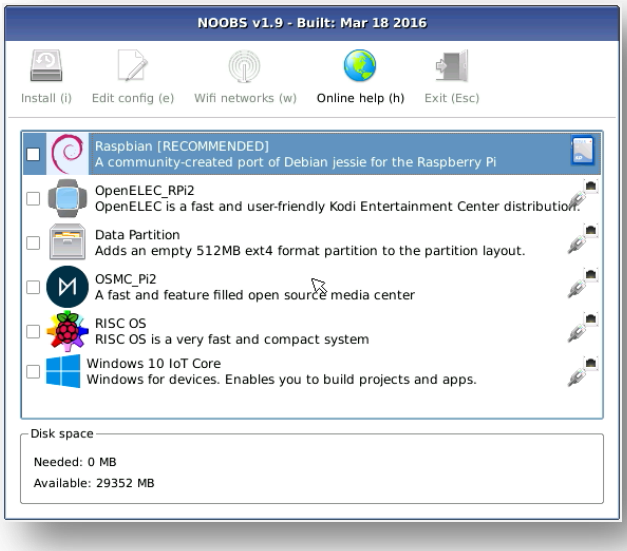


Figura 74. Menú de instalación de NOOBS.

Una vez introducida la tarjeta microSD en el puerto correspondiente de la Raspberry hará falta conectar los **cables de red**, de **salida HDMI** y de **alimentación, y el ratón y teclado** para proceder a la instalación de los S.O. desde el menú de NOOBS:

Se trata del **menú de instalación de los S.O.** En este menú se seleccionarán los S.O. que se quieren instalar, entre ellos **Raspbian**, ya que si se instala otro posteriormente, los datos de los programas previamente instalados se borrarán. Esta operación tardará varios minutos, en función de los S.O. que se vallan a instalar inicialmente. Hay que tener en cuenta que al instalar un S.O. se ocupa parte de la memoria de la microSD, dato que hay que tener en cuenta si se dispone de una SD de poca capacidad, por eso se recomienda una memoria de 16GB.

Para **instalar Raspbian** hay que **seleccionarlo** en el menú de NOOBS y darle al **botón install**, con ello comenzará la instalación.

Se podrá **acceder al menú** de **NOOBS al pulsar la tecla Shift** del teclado cada vez que se **inicia la Raspberry**.

4.2.5.2. Configuración de Raspbian

Una vez instalado Raspbian en la aplicación NOOBS ejecutaremos su archivo para entrar dentro del S.O. Una vez elegido este se iniciará junto con la Raspberry excepto que se pulse la tecla shift durante la iniciación de NOOBS.

Al entrar en Raspbian, aparecerá el escritorio del interfaz gráfico. En primer lugar hay que **modificar la configuración de Raspbian**, ya sea entrado a través del menú, en configuración Raspberry Pi, o mediante el terminal con la orden raspi-config.

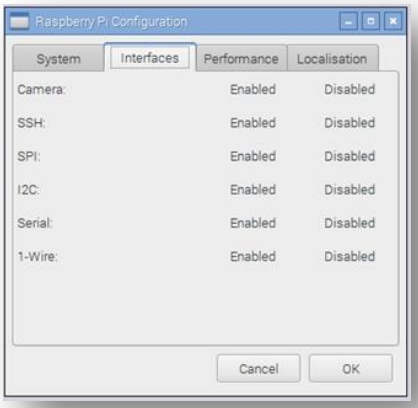


Figura 75. Menú de configuración del interfaz gráfico de Raspbian.

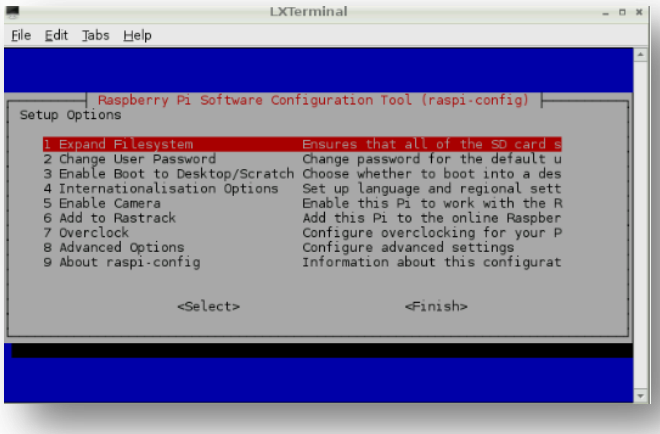


Figura 76. Menú raspi-config.

En este menú de configuración se podrá **modificar la contraseña** de la Raspberry, empleada para el acceso mediante SSH por ejemplo, y **se habilitará la cámara**. Tras realizar los cambios será necesario reiniciar el sistema.

Se recomienda realizar esta modificación a través del menú del interfaz gráfico, ya que a su vez se va a poder cambiar el idioma, la hora y el tipo de teclado.

4.2.5.3. Conexión de los periféricos

Antes de volver a encender la Raspberry, se va a realizar la **conexión de los periféricos**. Se recomienda realizar esta conexión con la Raspberry apagada, ya que si está encendida los periféricos no comenzarán a actuar hasta que esta se reinicie.

La **cámara se conectará al puerto de salida correspondiente**, como se ha comentado en el apartado 2.5.2. Los **servos** irán conectados **al puerto GPIO**, en este caso los **cables marrones y rojos a tierra y 5V respectivamente**, y los **cables naranjas a los puertos GPIO 18 y GPIO 19 a través de una resistencia de 1 KΩ**. El **servo** encargado del **giro** se conecta al **puerto 19**, y el encargado de la **elevación al 18**.

Las **conexiones de Ethernet** y de **salida HDMI se han conectado previamente** para la instalación de los S.O. en NOOBS. La salida HDMI ya no será imprescindible, ya que se podrán emplear sistemas de acceso o visualización remota como PuTTy o VNC viewer. Además la desconexión de la salida HDMI evitará problemas de alimentación al reducir el consumo.

4.2.5.4. Instalación de los programas

El último paso que hay que realizar es la instalación de los programas. Pero antes de ello será necesaria la **actualización del S.O**. Con ello se **evitarán problemas** de incompatibilidad entre el S.O. y los programas y periféricos. Esto puede ocurrir ya que ciertos programas que se instalan los actualizan para las últimas versiones de Raspbian.

Por tanto en el terminal se introducirán los siguientes comandos:

- sudo apt-get update
- sudo apt-get upgrade

Una vez actualizado el S.O. y los programas que vienen preinstalados en él, se procederá a la instalación de los demás programas.

Para poder **emplear interfaz gráfico remoto** se va a instalar en primer lugar **VNC server**:

- cd
- sudo apt-get install tightvncserver

A continuación se va a instalar **MJPEG-Streaming**. Para ello se requieren los siguientes comandos:

- cd
- sudo apt-get install libjpeg8-dev imagemagick libv4l-dev
- sudo ln -s /usr/include/linux/videodev2.h /usr/include/linux/videodev.h
- wget http://sourceforge.net/code-snapshots/svn/m/mj/mjpg-streamer/code/mjpg-streamer-code-182.zip
- unzip mjpg-streamer-code-182.zip
- cd mjpg-streamer-code-182/mjpg-streamer
- make mjpg_streamer input_file.so output_http.so
- sudo cp mjpg_streamer /usr/local/bin
- sudo cp output_http.so input_file.so /usr/local/lib/
- sudo cp -R www /usr/local/www

Tras ello se instalará **RPI WEB CAM INTERFACE**

- cd
- git clone https://github.com/silvanmelchior/RPi_Cam_Web_Interface.git
- cd RPi_Cam_Web_Interface
- chmod u+x RPi_Cam_Web_Interface_Installer.sh
- chmod u+x start.sh
- chmod u+x stop.sh
- ./RPi_Cam_Web_Interface_Installer.sh install
- ./RPi_Cam_Web_Interface_Installer.sh autostart_yes

Para la instalación del **visor VLC** se emplearán los siguientes comandos:

- cd
- sudo apt-get install vlc

Con esto queda realizada la instalación de los programas de video en *streamig* y se prodecerá con la instalación de **Weaved** y **Dropbox**. Para configurar **Weaved** se escribirán los siguientes comandos en el terminal:

- cd

- wget https://github.com/weaved/installer/raw/master/binaries/weaved-nixinstaller_1.2.13.bin
- chmod +x weaved-nixinstaller_1.2.13.bin
- ./weaved-nixinstaller_1.2.13.bin

Y para la instalación de **Drpobox**:

- cd
- git clone https://github.com/andreafabrizi/Dropbox-Uploader.git
- cd Dropbox-Uploader
- ./dropbox_uploader.sh

Como se ha podido observar, la instalación de todos los programas de realiza mediante el terminal, por ello se recomienda realizar estos pasos a través de un **SSH** como **PuTTy**, ya que se podrán copiar los comandos de cualquier documento y pegarlos directamente al SSH, evitando así posibles problemas de escritura y simplificando la instalación.

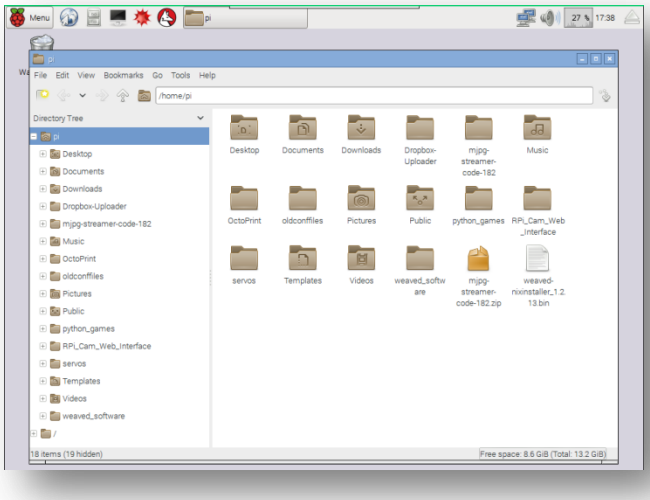


Figura 77. Menú principal de directorios.

El comando cd dirigirá la instalación al **directorio principal**, en el cual, una vez realizada la instalación, **aparecerán los subdirectorios de todos los programas instalados**:

4.2.5.5. Implementación de RPI WEB CAM INTERFACE

Tras realizar el estudio de los diferentes programas de visualización de video en *streaming* se ha concluido con la **elección de RPI WEB CAM INTERFACE**, sin embargo hay que conseguir **implementar** el programa para **introducir la configuración de los servos** dentro de éste.

Para poder realizar esta modificación se va a realizar un estudio más a fondo de este programa, para conocer la función de cada archivo y la programación empleada.

En primer lugar hay que fijarse que los archivos de este programa aparecen divididos, y algunos repetidos, en dos directorios.

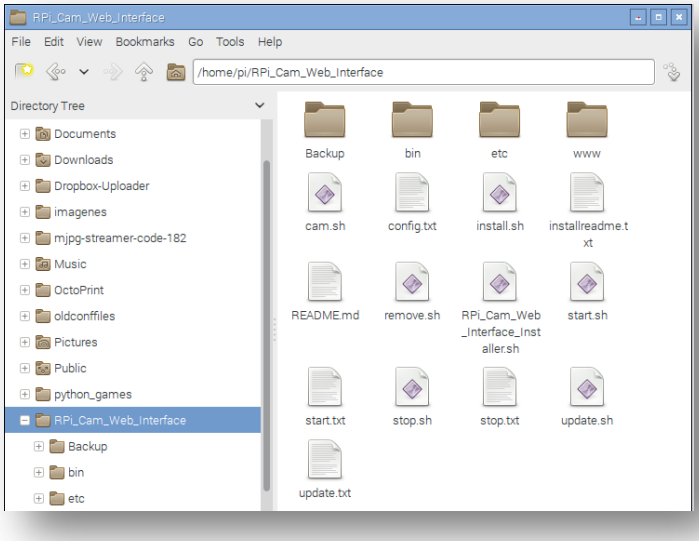


Figura 78. Directorio RPi_Web_Cam_Interface.

El primero de ellos, el **directorio Rpi_Web_Cam_Interface**, en el que aparecerán los **ficheros de inicio, parada, eliminación e instalación**, y será desde donde se **controle el programa** y se ejecute.

Sin embargo, es en el **segundo directorio** en el que se encuentra la **programación principal**. Este fichero se encuentra en **var/www/html**, y se trata del **directorio empleado por el servidor web Apache**. Es de este directorio donde hay que **introducir los programas que queremos que emplee el servidor Apache para la obtención de la página web.**

Este servidor es capaz crear páginas web dinámicas que utilizan lenguaje PHP. Por tanto se va a esperar que el lenguaje empleado por el programa RPI WEB CAM INTERFACE sea PHP.

En este directorio se van a encontrar los siguientes archivos y subdirectorios:

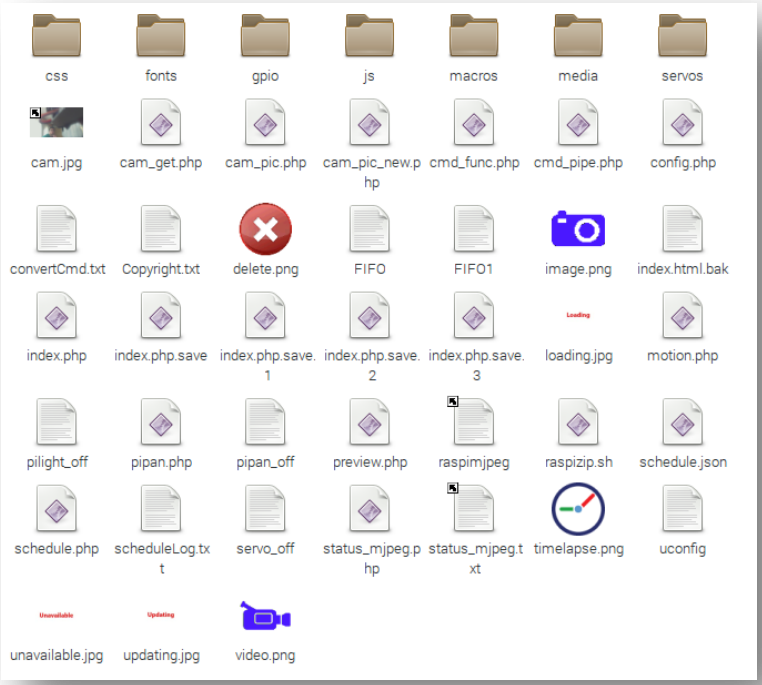


Figura 79. Archivos del directorio /var/www/html.

Se va a hacer una breve descripción de los **archivos más importantes** y sobre los cuales se basa el programa.

- **Config.php**

En la programación de config.php **se definen los elementos que van a emplearse en el programa.** Entro ellos destacan:

  - Nombre de la aplicación, host y cámara.
  - Se define raspimeg como archivo en el que se guardan los cambios de la configuración por defecto.
  - Se define uconfig como archivo en el que se guardan los cambios en la configuración específica del usuario.
  - Se define media como directorio en el que se almacenarán los archivos obtenidos con el programa, es decir, fotos y videos.
  - Define scheduleLog.txt como archivo de los registros de inicio de sesión

- **Index.php**

En este archivo se encuentra la **programación principal que configura el programa y la distribución de la página web.**

Se podrá observar que este programa posee **3 modos de funcionamiento**, uno de ellos para el **empleo de servos**, a los cuales se accede creando archivos llamados servo_on (modo 2) o pipan_on (modo1). Sin embargo **no se han obtenido los resultados esperados** con estas variables, ya que **no se ha conseguido el funcionamiento de los servos**, y se ha desestimado su empleo. Esto se cree que es debido a que el programa adicional que se emplea para ello, servoblaster, no tiene una versión que funcione bien en la Raspberry Pi 2 y Raspbian.

Este archivo se ha introducido como anexo debido a su extensión, y se podrá encontrar en **Anexo 2. Index.php**.

Por tanto, y como se ha explicado en el apartado 3.2. de lenguaje de programación PHP, **la programación comenzará con una declaración del tipo de documento,** en este caso **<!DOCTYPE html>,** es decir, versión 4.01 de html. A continuación vendrán los **delimitadores**, en primer lugar **de PHP**, y tras él los de **encabezado** y **cuerpo** en los que se configurará la página web.

- **Delimitador PHP**

Tras él se **declaran las variables** que se van a emplear en el programa.

A continuación se definen funciones para los modos de funcionamiento con pipan y pilight, los cuales no se van a emplear, y por tanto no se explicarán en detalle.

Se **definen** ciertas **funciones** como makeImput, getImgWidth o getLoadClass.

Se **definen los botones Simple** y **MJPEG-streamer y su función**, los cuales se podrán observar en la página web.

- **Delimitador head**

En este apartado se introducirá la programación correspondiente al **encabezado**. En él se introducirá el **título** y se define el **estilo** del mismo.

- **Fin delimitador head**
- **Delimitador body**

En este delimitador se escribirá el **cuerpo del programa**. Para ello se introducirán las **divisiones, saltos de línea y botones correspondientes**.

En primer lugar se introduce el **botón Single** que cambiará de vista simple a vista con botones y desplegables, **seguido de la imagen**, la cual se introducirá a mediante las funciones getImgWidth y getLocalClass y la dirección de la imagen mjpeg_dest.

Tras ello, se escriben los **comandos relacionados con los botones de captura de video, captura de imagen, timelapse, motion y parada.**

A continuación de definen los **botones de descarga de videos e imágenes**, que enlazará con la pagina creada en preview.php, de **editar la configuración de motion** y de **editar la configuración de los registros.**

Se crea un **desplegable** con el título **configuración de la cámara**, en la que aparecerá una **tabla con los parámetros de la cámara**, y dará opción a modificarlos. Para crear esta tabla se emplearán los indicadores tr, td y th descritos anteriormente.

Estos parámetros modificarán el valor de las variables definidas al inicio del programa, exceptuando la resolución, que se introduce directamente en la función makeImput comentada anteriormente.

Tras finalizar esta tabla, se va a crear otra para el apartado de **configuración de motion**, que se creará de igual manera que la primera tabla.

Por último se introducirán los **botones de MJPEG streamer**, de **reinicio**, de **apagar** el sistema y de **restablecer la configuración inicial**.

- **Fin delimitador body**
- **Fin delimitador PHP**

Con esto queda definida la página web y su estilo, con la introducción de la imagen y los botones y tablas.

- **Preview.php**

En este archivo se crea la **configuración de la página correspondiente a las descargas de vídeo e imágenes.**

En primer lugar se **configura el acceso**, que se trata del **botón de descarga de config.php**. Seguidamente **se definen los botones y textos que se emplearán en el programa**. Entre ellos se encuentra **el botón convertir**, que emplea la **función escrita en convertCmd.txt**.

A continuación de definen el **tamaño de la visualización de las imágenes** y las **funciones de los botones.**

Tras ello se define el tipo de documento **<!DOCTYPE html>** y se escribe el **texto PHP junto con el encabezado y el cuerpo**. En el **cuerpo** se **definirá la configuración de los botones y las miniaturas de las imágenes.**

- **Schedule.php**

Contiene la **configuración de la página de edición de registros a la que se accede desde el botón correspondiente de la programación config.php.** Como no se van a modificar los registros no se va a realizar la descripción de este archivo.

- **Raspimjpeg**

Contendrá la **información de la configuración por defecto**. Por ello almacenará el valor de todas las variables, como el de los parámetros de la imagen, el ancho o calidad de video en directo, la configuración de motion, la localización de los archivos, etc.

- **Uconfig**

En este archivo **se guardarán los datos correspondientes a cambios en la configuración específica del usuario**, como por ejemplo si se realizan cambios en los parámetros de la cámara.

Una vez comprendida la programación de RPI WEB CAM INTERFACE, se va a poder añadir una rama de programa en la que **introducir el funcionamiento de los servos**.

Como ya se ha comentado en el apartado 4.2.2. de los servomotores, ya que no se ha encontrado la manera de exportar datos de un programa en PHP para introducirlos en otro programa ajeno escrito en Python, se ha optado por **crear pequeños programas en Python a los que se acceda mediante la pulsación de botones.**

Para ello se han realizado diversos programas en Python con la configuración básica para el movimiento de los servos, con un ángulo ya fijado, por tanto no dependerá de ningún valor introducido mediante el navegador, sino que en función del botón pulsado se ejecutarán distintos programas de Python.

Para introducirlos se han empleado las siguientes instrucciones en PHP:

```
<!--GPIO18-->
<form action="" method="post">
```

```
GPIO 18 
<input type="submit" name="nombre" value="Valor">
```

Siendo *nombre* el nombre que se le quiere asignar al botón, y *valor* el valor que le asigna a la variable al pulsarla.

En esta primera parte se **inicializan los puertos GPIO** correspondientes a los servos en lenguaje PHP y se define el botón, su nombre y valor de salida.

```
<?php
  if ($_POST[value]) {
        for($i=0, $i=10,$i++){
            $a- exec("sudo python /var/www/html/servos/gpio/18/programa.py");
            echo $a;
        }
   }
?>
```

En la segunda parte se crean **bucles que ejecutarán el programa** de Python elegido tras la pulsación de un botón.

El bucle for se ha añadido debido a que el tiempo de ejecución del programa en Python era muy pequeño y no daba tiempo a establecer el valor requerido por el usuario al servo. Por ello se repite la acción cierto número de veces.

Esta parte es la que contiene la información html, por ello va delimitada con los correspondientes indicadores.

Los programas Python empleados para esta función siguen la siguiente estructura:

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
GPIO.setup(18, GPIO.OUT)
pwm = GPIO.PWM(18, 20)
angle=0
duty = angle / 56 + 3.5
for i in range (1,50):
      pwm.start(duty)
```

Se configura el **puerto 18 o 19** del puerto GPIO como **salida**, con **frecuencia de 20kHz**, periodos de refresco cada 50ms, y se inicia el **PWM** con cierto ciclo de trabajo. En este caso los ciclos de trabajo se han realizado de forma teórica añadiendo un ajuste final de los valores de forma práctica tras probarlos.

El bucle for en el que se han introducido es debido a que si no se ejecutaba el programa rápidamente y no le daba tiempo a alcanzar la posición deseada al servo, sin embargo posteriormente se vio que este error era debido al programa PHP y no a este. Aun y todo no se han modificado ya que de esta manera el programa funciona correctamente.

Podría implementarse este programa mediante la orden **GPIO.cleanup()**, que limpia la información de los puertos empleados. Este método se ha empleado para la prueba de programas en el escudo de Pyhon IDLE, en el cual se mejoraba respecto a rebotes o movimientos inesperados. Sin embargo, respecto a la lectura de dichos programas mediante PHP no hay ninguna diferencia en cuanto su empleo.

Se han introducido en el programa los siguientes **valores para la posición de los servos**:

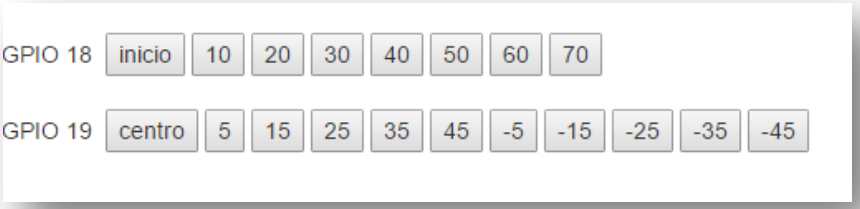


Figura 80. Botones referentes al movimiento de los servos.

Se ha optado por escribir estos comandos al inicio del delimitador body para que aparezcan **al inicio del programa**.

Los programas Python para el movimiento de los servos de encuentran en var/www/html/servos/gpio, en los directorios 18 y 19, en función de que servo se trate. Para escribir los programas en esta carpeta se necesita permiso de raíz, y por tanto una vez escrito el programa será difícil borrarlo por equivocación.

Tanto para escribirlos como para modificarlos se ha empleado mediante el terminal, el editor de texto nano junto con el comando de permiso de raíz sudo.

Con esto se daría por finalizado el proyecto, habiendo obtenido los resultados esperados, y **habiendo cumpliendo con los objetivos** iniciales del mismo. El programa desarrollado posee la programación necesaria para realizar el **correcto movimiento de los servomotores** y permite la **visualización de vídeo en directo**, a la vez que permite **obtener y guardar imágenes y vídeos**. El **acceso** a esta página web se realiza **mediante internet**, sin ser necesaria la conexión por red local, por lo que es accesible para cualquier usuario en cualquier lugar.

4.2.5.6. Soportes

Para poder **instalar la Raspberry junto con la cámara y servos en la impresora 3D** se ha realizado un **soporte** que albergue el conjunto y que sea fácilmente desplazable. La idea inicial era crear un soporte para la **grabación** de la impresora 3D **desde la parte frontal**.

Para ello se ha realizado un soporte que iba a estar sujeto a una de las rendijas frontales de la impresora 3D, en el que se colocará la cámara montada en la plataforma de los servos en la parte opuesta a la sujeción.

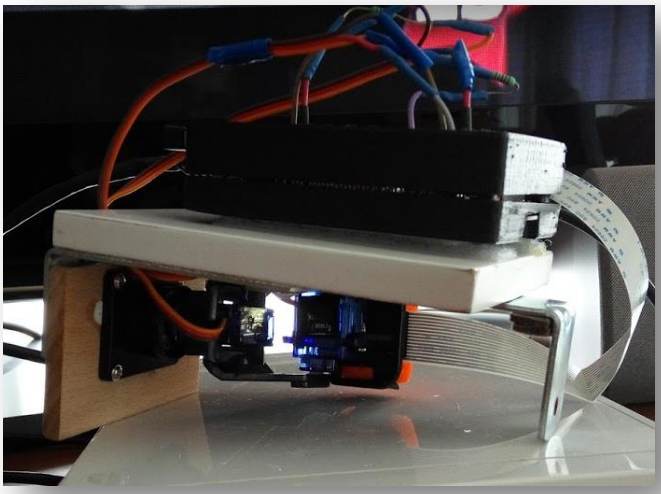


Figura 81. Soporte para la Raspberry, servos y cámara.

Esta montura se ha realizado mediante dos maderas y 3 escuadras, una de soporte y 2 para la unión de las maderas. La Raspberry se une a la madera superior por medio de velcros, para facilitar así la separación de la misma es caso de requerirse. En cuento a los servos, se han unido a través de 4 tornillos a la madera trasera.

Una vez probado el soporte, se ha tenido que **modificar su ubicación**, debido a la dificultad de grabar la parte superior de la impresora 3D. Esto ocurre ya que para permitir la apertura de la puerta no se puede bajar la cámara lo necesario para poder enfocar dicha parte superior.

La **ubicación final** del soporte será **sobre la impresora**, **atornillada a la carcasa** de la misma a través de la escuadra de soporte de la montura.

A su vez, **para proteger la cámara** de golpes se ha imprimido una **carcasa** mediante la impresora 3D:

Figura 82. Funda impresa para la cámara.

El enlace al archivo de impresión es el siguiente:

http://www.thingiverse.com/thing:92208

Una vez atornillado el soporte en la impresorá, el resultado obtenido es el siguiente:

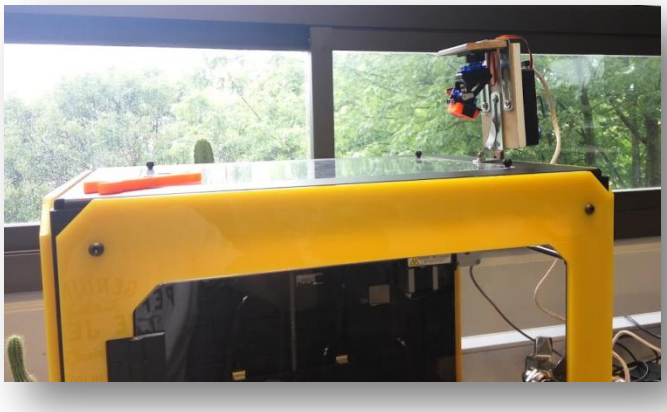


Figura 83. Instalación final del soporte

Figura 84. Instalación final del soporte

Y las imágenes obtenidas mediante la cámara tendrán el siguiente enfoque:

Figura 85. Imágen final de la visualización de vídeo en streaming

5. Problemas

Durante en desarrollo del proyecto, han surgido varios problemas que ha habido que solucionar para poder obtener un correcto funcionamiento de la Raspberry. En este apartado se van a remarcar los problemas más importantes, puesto que los demás se han ido solucionando y explicando durante la memoria.

5.1. Problema de alimentación

Una de los mayores problemas que se han encontrado es la de **obtener una alimentación adecuada** para la Raspberry. Inicialmente se ha empleado un cargador con salida de 1000mA y 5V, suficiente según las especificaciones técnicas, que sugieren una alimentación de 800mA. Sin embargo, a medida que se le conectan los periféricos, se incrementa la demanda de corriente de la Raspberry, provocando errores y reiniciándose la misma.

Para solucionarlo, se ha cambiado el transformador del cargador, a uno con **salida 2000mA y 5V**, permitiendo conectar a la vez la cámara, los servos, el ratón y teclado, la salida HDMI y empleando los programas desarrollados sin ningún problema.

Por tanto se puede decir, que la alimentación optima de la Raspberry se sitúa en valores de corriente de 2000mA para este proyecto.

5.2. Incompatibilidad de programas

Uno de los problemas que más se han repetido a lo largo del proyecto es la **incompatibilidad de los programas descargados con el S.O.** Esto se debe a que hay momentos en los que la **versión** de los programas descargados es más reciente que la última actualización de nuestro S.O. y por tanto **no se corresponden**, o que dicho programa descargado emplee otro programa ya instalado en la Raspberry pero no actualizado a la versión que este emplea. Los problemas generados por la incompatibilidad son difíciles de distinguir, ya que se **producen fallos** en el programa descargado, pero no se sabe de donde provienen.

Para **solucionarlo** se recomienda **actualizar el sistema** con las órdenes siguientes tras instalar el S.O., y si es necesario antes de instalar cada programa:

```
sudo apt-get update

sudo apt-get upgrade
```

5.3. Permiso de administrador

A la hora **de modificar o crear programas existentes** en la Raspberry, en función del directorio en el que se encuentren se **requerirá permisos de raíz o administrador**. Estos archivos se encontrarán en la carpeta ./, en la cual se puede acceder desde el administrador de directorios. En este directorio se almacenan los archivos que emplean la Raspberry y ciertos programas para poder funcionar correctamente.

Para poder acceder a estos archivos o crear nuevos, se empleará la **orden sudo del terinal**, que **adjudica los permisos de administrador al usuario**. Con esta orden y la ayuda de un editor de texto como nano se podrán modificar los programas.

5.4. Conexión a internet

La **conexión a internet de la universidad no permite conectar a la red dispositivos externos** que no hayan recibido permiso de conexión. En cuento a la conexión por Wifi, la red no permite la exportación de archivos ni datos, por tanto no servirá para realizar el proyecto.

Para **solucionar** este problema ha hecho falta **dar de alta la dirección mac de la Raspberry** en la red de la universidad, y los puertos en los que esta va a ser empleada. Así se ha suministrado a la Raspberry una **dirección fija en la red de la universidad** y se le ha proporcionado acceso desde los terminales que va a emplearse.

La dirección IP asignada a la Raspberry es la siguiente:

- Dirección IP: 172.18.93.121
- Mascara de Subred: 255.255.224.0
- Puerta de enlace: 172.18.64.254
- DNS: 130.206159.2

6. Conclusiones

La primera conclusión que hay que destacar es la **consecución de los objetivos iniciales** del proyecto, ya que finalmente, mediante el empleo de la Raspberry Pi 2B y el programa RPI WEB CAM INTERFACE implementado se ha conseguido cumplir los requisitos propuestos.

Sin embargo en este proyecto se ha intentado ir más allá de la consecución de los objetivos iniciales, y se ha tratado de dar unas **pautas de iniciación al lector para emplear la Raspberry Pi,** y la información básica de la programación que emplea el S.O. Raspbian. Con ello se pretende fomentar la utilización de esta placa de programación, con la que se pueden realizar grandes proyectos.

La Raspberry proporciona al usuario una plataforma en la que poder obtener programas relacionados a cualquier campo tecnológico, ya sea la programación, el almacenamiento de datos o emplear la Raspberry como videoconsola o centro multimedia. Bastará con emplear los distintos S.O. disponibles, obteniendo resultados sorprendentes.

En lo referente al uso de la Raspberry para la programación, la **elección del S.O. Raspbian** ha sido un acierto, ya que al inicio el poder emplear el **interfaz gráfico** simplifico mucho la interacción con la Raspberry. Además la **mayoría de información** encontrada en relación a los objetivos del proyecto era **referente a este S.O.** Esto ha sido una gran ventaja a la hora de encontrar un programa válido para este proyecto, además de tener gran variedad de opciones.

En cuanto a la programación, se eligió **Python** sobre C como base inicial de la programación para familiarizarse con la Raspberry. A pesar de tener conocimientos previos sobre la programación en C, se ha empleado Python, y se han adquirido suficientes conocimientos para la realización de programas. Python es una **gran elección en cuanto al S.O. Raspbian**, debido a su **simplicidad**, el **empleo de muchas librerías** referentes a diversos campos, y permite crear **funciones y programas sencillos** pero de gran funcionalidad. En este proyecto se han empleado las librerías Tkinter, picamara, time y socket, las cuales permiten realizar programas complejos mediante ordenes sencillas.

Centrándonos en los programas probados, **RPI WEB CAM INTERFACE** ha resultado ser el **programa más funcional**, ya que **permite obtener imágenes y vídeos** de la **visualización en streaming**, almacenarlas, descargarlas e incluso modificar parámetros de la cámara. Todo esto lo permite realizar de **manera sencilla**, accediendo a través de botones a las funciones. A su vez, si se quiere emplear este programa para cámara de seguridad, posee la extensión motion, de gran funcionalidad en este apartado.

7. Líneas futuras

En cuanto a las **líneas futuras** de este proyecto se quiere destacar el **control de la impresora 3D** mediante la Raspberry. Se ha realizado un pequeño estudio de este apartado a través del **programa Octoprint**.

Octoprint es un programa diseñado para el control de impresoras 3D mediante la Raspberry. Se trata de un programa gratuito, por tanto se ha podido probar el funcionamiento del programa.

Se trata de un programa muy interesante, ya que entre sus **funciones** destacan:

- La medida de temperatura de la impresora
- Permite introducir archivos de impresión GCode y mandar la orden de impresión.
- El control de la base y ejes de la impresora, permitiendo modificar su posición.
- La visualización de video en *streaming*, y la obtención de *timelapses* para crear videos de las impresiones.
- Permite ver el archivo GCode y cómo va a ser su realización.
- Se le pueden introducir órdenes a través de un terminal para la modificación de parámetros.

La **instalación y puesta a punto** de Octoprint es bastante sencilla, puesto que realizará de la misma manera que los demás programas. En lo referente a la instalación, esta se realizará por medio del terminal:

```
cd

sudo apt-get install python-pip python-dev git
```

Se instalan las dependencias básicas

```
git clone https://github.com/foosel/OctoPrint.git
```

Se descarga el programa

```
cd OctoPrint

sudo pip install -r requirements.txt
```

Se instala las dependencias.

Una vez instalado, para **ejecutarlo** hace falta introducir el **comando octoprint** en el terminal. Una vez arrancado el programa, hay que dirigirse al **puerto 5000** de la Raspberry con ayuda de un navegador.

Una vez en la página web, se nos pedirá por un usuario y contraseña que habrá que establecer en ese momento.

Los archivos GCode se podrán subir desde la ventana principal y quedarán guardados en la memoria de la Raspberry. A su vez, desde el menú principal se iniciará la conexión entre impresora y Raspberry, las cuales se conectarán mediante un bus de datos con salida USB desde la Raspberry.

En cuanto a la visualización de vídeo en directo, habrá que realizar la configuración del mismo. Una opción encontrada tras varias pruebas, fue emplear la **imagen obtenida por el programa MJPEG-Streamer.** Para ello se emplea la dirección URL del vídeo que proporciona este programa, y se introduce en la **configuración de vídeo** de Octoprint.

Stream URL : http://192.168.1.41:8080/?action=stream

Snapshot URL : http://192.168.1.41:8080/?action=stream

Path to FFMPEG : /tmp/stream/pic.jpg

Se ha añadido también el camino que sigue el programa MJPEG-Streamer para la imagen empleada en el vídeo. Este es un ejemplo para la dirección empleada en la red local en la que estaba instalada la Raspberry. Para que funcione en la universidad hará falta cambiar dicha dirección por la correspondiente.

Con la introducción del video, aumentarán las órdenes que deben ser introducidas en el terminal. Deberán **emplearse 2 ventanas de terminal**, una para los **comandos de octoprint** y otro para los de **MJPEG-Streamer**. Los comandos a emplear son los siguientes:

```
Mkdir /tmp/stream
raspistill --nopreview -w 640 -h 480 -q 85 -o /tmp/stream/pic.jpg -tl 1 -t 9999999 -th 0:0:0 &
LD_LIBRARY_PATH=/usr/local/lib mjpg_streamer -i "input_file.so -f /tmp/stream -n pic.jpg" -o "output_http.so -w /usr/local/www"
octoprint
```

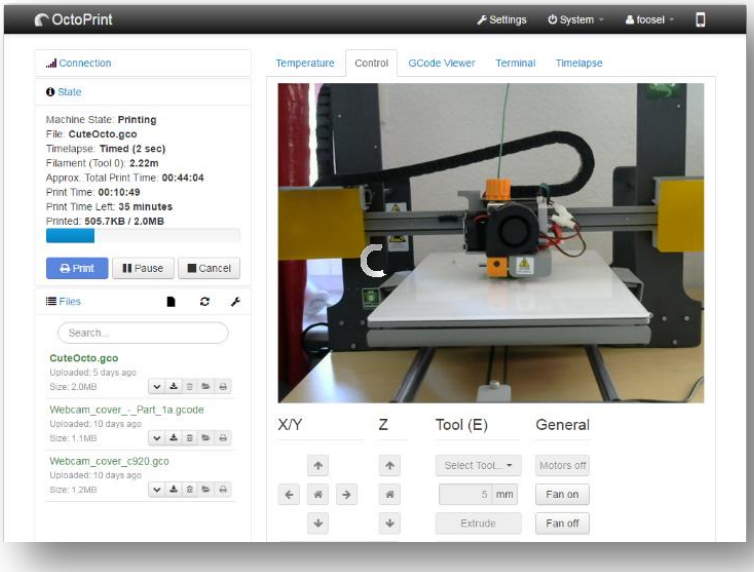


Figura 86. Ventana de control del programa Octoprint. http://octoprint.org/

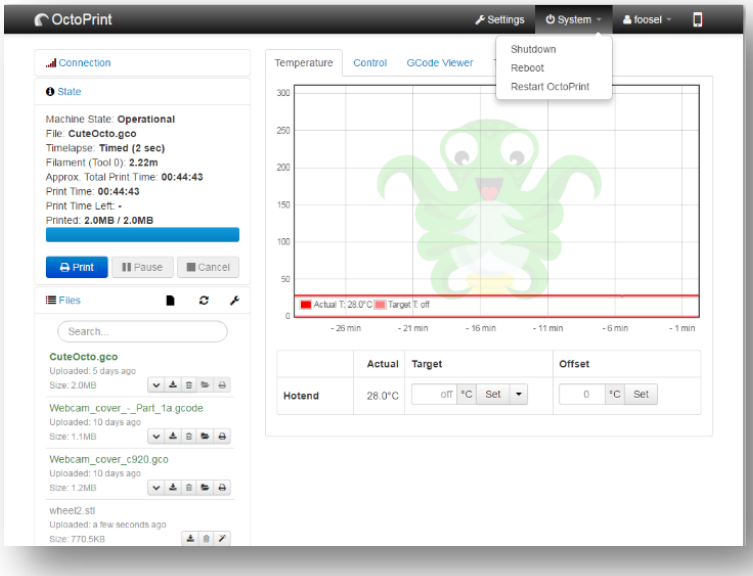


Figura 87. Ventana de temperatura y desplegable del sistema del programa Octoprint. http://octoprint.org/

En el anexo 2 se explican los principales comandos que se pueden emplear en el terminal de octoprint y se proporciona un enlace para la explicación de la función de cada botón de la página web octoprint.

Con esto **Octoprint** se postula como la **principal línea futura** del proyecto, sin embargo hay otras opciones para este ámbito. Una de las propuestas es la **implementación de RPI WEB CAM INTRFACE con Dropbox.**

Para ello hay que **introducir en el archivo preview.php** del programa un **cuadro de texto en el que se ejecute la orden de Dropbox** con la imagen seleccionada. Su funcionamiento puede basarse en la orden convertirCMD que emplea para convertir las imágenes *timelapse* en vídeo.

Por otro lado, este programa viene predefinido para controlar servos, sin embargo necesita del **programa servoblaster**, el cual ha dado **problemas** para instalarse en esta versión de Raspbian. Otra línea futura se basa en la obtención de dicho programa para la versión empleada de Raspbian y mediante la configuración pipan_on y servo_on comentada en el apartado 4.2.5.5. , conseguir un **control más preciso de los servomotores.**

Para no depender de una conexión fija a internet y no necesitar de cable Ethernet, es posible emplear un **módulo Wifi** conectado a un puerto USB. La configuración del Wifi puede realizarse mediante el programa NOOBS si se desea hacer la descarga de los S.O. con Wifi, o si no se puede configurar desde el menú de configuración de Raspbian.

Las pautas a seguir para la instalación del Wifi se encuentran en el apartado 4.1.2. Conexión a internet. Sin embargo, hay que tener en cuenta que en la red de la universidad no funcionará el acceso mediante Wifi para la exportación de datos.

Por otro lado se ha querido acercar al usuario al entorno de trabajo de Raspberry, por lo que se anima a probar la **creación de programas** o descubrir nuevos programas para este S.O. u otro, como puede ser Window 10 IoT, el cual está ganando muchos adeptos.

8. Referencias

Información general:

https://raspberryparatorpes.net
http://nergiza.com/raspberry-pi-2-b-que-es-y-para-que-nos-puede-servir/
http://rootear.com/tag/raspberry-pi/
http://dplinux.net/guia-raspberry-pi/#__RefHeading__9664_924516217
José Andrade: Raspberry Pi modelo B analizado. www.engadget.com:
http://es.engadget.com/2012/08/11/raspberry-pi-model-b-analizado/

Sistemas operativos:

http://comohacer.eu/distribuciones-raspberry-pi/
https://raspberryparatorpes.net/raspberry-pi-sistemas-operativos/
http://www.informatica-hoy.com.ar/aprender-informatica/Que-es-el-sistema-operativo.php
https://www.masadelante.com/faqs/sistema-operativo

Lenguajes de programación:

http://definicion.de/lenguaje-de-programacion/

Raspbian:

http://comohacer.eu/distribuciones-raspberry-pi/
http://www.neoteo.com/raspbian-el-sistema-operativo-del-raspberry-pi

Windows 10 IoT:

http://www.hwlibre.com/como-instalar-windows-10-en-tu-raspberry-pi-2/#Descargar_el_ISO_de_Windows_10_IoT
http://www.techrepublic.com/article/windows-10-on-the-raspberry-pi-what-you-need-to-know/
http://www.redeszone.net/2015/08/15/windows-10-iot-core-para-el-raspberry-pi-2-ya-se-encuentra-disponible/#sthash.4y7UxBaI.dpuf
http://ms-iot.github.io/content/en-US/IoTCore.htm

Python:

https://librosweb.es/libro/algoritmos_python/
http://docs.python.org.ar/tutorial/pdfs/TutorialPython2.pdf
https://docs.python.org/3/tutorial/index.html
http://librosweb.es/libro/python/capitulo_13/python_bajo_apache.html

PHP:

http://servicio.uca.es/softwarelibre/publicaciones/apuntes_php
http://php.net/

http://www.desarrolloweb.com/php/

Visual Studio:

https://www.visualstudio.com/

PuTTy:

http://www.putty.org/
http://es.ccm.net/download/descargar-395-putty

VNC Server:

https://www.realvnc.com/download/viewer/
https://wiki.archlinux.org/index.php/TigerVNC_(Espa%C3%B1ol)
https://www.raspberrypi.org/documentation/remote-access/vnc/

Win32 Disk Imager:

http://win32-disk-imager.uptodown.com/windows

SDFormatter:

https://www.sdcard.org/downloads/formatter_4/

Advanced IP Scanner:

http://www.advanced-ip-scanner.com/es/

Cámara:

http://picamera.readthedocs.org/en/release-1.2/recipes1.html
https://www.raspberrypi.org/documentation/hardware/camera.md
https://docs.google.com/document/d/1e7U_WM1jSo0wgMX5RdpN0eN6p9TWadGYHqWf6yR2JZM/edit

Pines GPIO:

http://www.raspberryshop.es/hardware-raspberry-pi.php
http://comohacer.eu/gpio-raspberry-pi/

Dropbox:

http://raspi.tv/2013/how-to-use-dropbox-with-raspberry-pi
https://github.com/andreafabrizi/Dropbox-Uploader

MJPG-STREAMER:

http://blog.miguelgrinberg.com/post/how-to-build-and-run-mjpg-streamer-on-the-raspberry-pi
http://datarhei.github.io/restreamer/docs/installation-linux-arm.html#raspberry-pi-2

motion:

http://www.scavix.com/files/ScavixSoftware_RaspberryPi_as_low-cost_HD_surveillance_camera_tutorial.pdf
www.youtube.com/watch?v=1yGi7_bWbFU

http://embeddedday.com/projects/raspberry-pi/a-step-further/install-motion-mmal/
https://www.maketecheasier.com/raspberry-pi-as-surveillance-camera/
https://geekytheory.com/twithief-twython-motion-playstation-eye-toy/

RPi_Cam_Web_Interface

https://github.com/skalad/RPi_Cam_Web_Interface_ServoBlaster_pan_tilt (tiene servos incluidos)
http://elinux.org/RPi-Cam-Web-Interface
https://github.com/skalad/RPi_Cam_Web_Interface_ServoBlaster_pan_tilt

VLC:

https://www.youtube.com/watch?v=JjPsW-7FUng
http://www.videolan.org/vlc/

Octoprint

http://octoprint.org/
https://github.com/foosel/OctoPrint
https://github.com/foosel/OctoPrint/wiki/Setup-on-a-Raspberry-Pi-running-Raspbian
http://enthings.com/software-3d/octoprint-la-suma-de-impresoras-3d-y-raspberry-pi/
http://spainlabs.com/wiki/index.php?title=Raspberry_pi_%2B_Octoprint
https://github.com/foosel/OctoPrint/wiki/Configuration
http://plugins.octoprint.org/
http://enthings.com/software-3d/octoprint-la-suma-de-impresoras-3d-y-raspberry-pi/

Conectarse a internet con otra IP:

http://readwrite.com/2014/06/27/raspberry-pi-web-server-website-hosting/

Weaved:

http://www.dexterindustries.com/howto/set-up-weaved-on-the-raspberry-pi/
https://developer.weaved.com/portal/members/iot_downloads.php
http://hacedores.com/raspberry-pi-weaved/
http://webiopi.trouch.com/

Servomotores:

http://fpaez.com/controlar-un-servomotor-con-raspberry-pi/
http://www.internetdelascosas.cl/2013/10/13/configurando-motion-con-la-camara-de-raspberry-pi/
https://www.youtube.com/watch?v=ddlDgUymbxc
http://razzpisampler.oreilly.com/ch05.html
http://www.toptechboy.com/raspberry-pi/raspberry-pi-lesson-28-controlling-a-servo-on-raspberry-pi-with-python/
https://geekytheory.com/tutorial-raspberry-pi-gpio-parte-2-control-de-leds-con-python/
http://raspberrypi.stackexchange.com/questions/12966/what-is-the-difference-between-board-and-bcm-for-gpio-pin-numbering

http://raspi.tv/2013/rpi-gpio-basics-4-setting-up-rpi-gpio-numbering-systems-and-inputs

Punto de acceso wifi:

http://willhaley.com/blog/raspberry-pi-hotspot-ew7811un-rtl8188cus/

http://www.pi-point.co.uk/documentation/

Anexo 1. Index.php

En este apartado se va a introducirla programación del archivo **index.php**, en el cual se ha introducido el movimiento de los servos. **Esta parte implementada queda destacada en color azul.**

```
<!DOCTYPE html>
<?php
  define('BASE_DIR', dirname(__FILE__));
  require_once(BASE_DIR.'/config.php');
  $config = array();
  $debugString = "";

  $options_mm = array('Average' => 'average', 'Spot' => 'spot', 'Backlit' => 'backlit', 'Matrix' => 'matrix');
  $options_em = array('Off' => 'off', 'Auto' => 'auto', 'Night' => 'night', 'Nightpreview' => 'nightpreview', 'Backlight' => 'backlight', 'Spotlight' => 'spotlight', 'Sports' => 'sports', 'Snow' => 'snow', 'Beach' => 'beach', 'Verylong' => 'verylong', 'Fixedfps' => 'fixedfps');
  $options_wb = array('Off' => 'off', 'Auto' => 'auto', 'Sun' => 'sun', 'Cloudy' => 'cloudy', 'Shade' => 'shade', 'Tungsten' => 'tungsten', 'Fluorescent' => 'fluorescent', 'Incandescent' => 'incandescent', 'Flash' => 'flash', 'Horizon' => 'horizon');
//    $options_ie = array('None' => 'none', 'Negative' => 'negative', 'Solarise' => 'solarise', 'Sketch' => 'sketch', 'Denoise' => 'denoise', 'Emboss' => 'emboss', 'Oilpaint' => 'oilpaint', 'Hatch' => 'hatch', 'Gpen' => 'gpen', 'Pastel' => 'pastel', 'Watercolour' => 'watercolour', 'Film' => 'film', 'Blur' => 'blur', 'Saturation' => 'saturation', 'Colourswap' => 'colourswap', 'Washedout' => 'washedout', 'Posterise' => 'posterise', 'Colourpoint' => 'colourpoint', 'ColourBalance' => 'colourbalance', 'Cartoon' => 'cartoon');
// Remove Colourpoint and colourbalance as they kill the camera
  $options_ie = array('None' => 'none', 'Negative' => 'negative', 'Solarise' => 'solarise', 'Sketch' => 'sketch', 'Denoise' => 'denoise', 'Emboss' => 'emboss', 'Oilpaint' => 'oilpaint', 'Hatch' => 'hatch', 'Gpen' => 'gpen', 'Pastel' => 'pastel', 'Watercolour' => 'watercolour', 'Film' => 'film', 'Blur' => 'blur', 'Saturation' => 'saturation', 'Colourswap' => 'colourswap', 'Washedout' => 'washedout', 'Posterise' => 'posterise', 'Cartoon' => 'cartoon');
  $options_ce_en = array('Disabled' => '0', 'Enabled' => '1');
  $options_ro = array('No rotate' => '0', 'Rotate_90' => '90', 'Rotate_180' => '180', 'Rotate_270' => '270');
  $options_fl = array('None' => '0', 'Horizontal' => '1', 'Vertical' => '2', 'Both' => '3');
  $options_bo = array('Off' => '0', 'Background' => '2');
  $options_av = array('V2' => '2', 'V3' => '3');
  $options_at_en = array('Disabled' => '0', 'Enabled' => '1');
  $options_ac_en = array('Disabled' => '0', 'Enabled' => '1');
  $options_ab = array('Off' => '0', 'On' => '1');
  $options_vs = array('Off' => '0', 'On' => '1');
  $options_rl = array('Off' => '0', 'On' => '1');
  $options_vp = array('Off' => '0', 'On' => '1');
  $options_mx = array('Internal' => '0', 'External' => '1');
  $options_mf = array('Off' => '0', 'On' => '1');
  $options_cn = array('First' => '1', 'Second' => '2');
  $options_st = array('Off' => '0', 'On' => '1');

  function initCamPos() {
    $tr = fopen("pipan_bak.txt", "r");
    if($tr){
      while(($line = fgets($tr)) != false) {
        $vals = explode(" ", $line);
        echo '<script type="text/javascript">init_pt(',$vals[0],',',$vals[1],');</script>';
      }
      fclose($tr);
    }
  }
```

```
function pan_controls() {
  $mode = 0;
  if (file_exists("pipan_on")){
    initCamPos();
    $mode = 1;
  } else if (file_exists("servo_on")){
    $mode = 2;
  }
  if ($mode <> 0) {
    echo '<script type="text/javascript">set_panmode(',$mode,');</script>';
    echo "<div class='container-fluid text-center liveimage'>";
    echo "<div alt='Up' id='arrowUp' style='margin-bottom: 2px;width: 0;height: 0;border-left: 20px solid transparent;border-right: 20px solid transparent;border-bottom: 40px solid #428bca;font-size: 0;line-height: 0;vertical-align: middle;margin-left: auto; margin-right: auto;' onclick='servo_up();'></div>";
    echo "<div>";
    echo "<div alt='Left' id='arrowLeft' style='margin-right: 22px;display: inline-block;height: 0;border-top: 20px solid transparent;border-bottom: 20px solid transparent;border-right: 40px solid #428bca;font-size: 0;line-height: 0;vertical-align: middle;' onclick='servo_left();'></div>";
    echo "<div alt='Right' id='arrowRight' style='margin-left: 22px;display: inline-block;height: 0;border-top: 20px solid transparent;border-bottom: 20px solid transparent;border-left: 40px solid #428bca;font-size: 0;line-height: 0;vertical-align: middle;' onclick='servo_right();'></div>";
    echo "</div>";
    echo "<div alt='Down' id='arrowDown' style='margin-top: 2px;width: 0;height: 0;border-left: 20px solid transparent;border-right: 20px solid transparent;border-top: 40px solid #428bca;font-size: 0;line-height: 0;vertical-align: middle;margin-left: auto; margin-right: auto;' onclick='servo_down();'></div>";
    echo "</div>";
  }
}

function pilight_controls() {
  echo "<tr>";
   echo "<td>Pi-Light:</td>";
   echo "<td>";
    echo "R: <input type='text' size=4 id='pilight_r' value='255'>";
    echo "G: <input type='text' size=4 id='pilight_g' value='255'>";
    echo "B: <input type='text' size=4 id='pilight_b' value='255'><br>";
    echo "<input type='button' value='ON/OFF' onclick='led_switch();'>";
   echo "</td>";
  echo "</tr>";
}

function getExtraStyles() {
  $files = scandir('css');
  foreach($files as $file) {
    if(substr($file,0,3) == 'es_') {
      echo "<option value='$file'>" . substr($file,3, -4) . '</option>';
    }
  }
}


function makeOptions($options, $selKey) {
  global $config;
  switch ($selKey) {
    case 'flip':
      $cvalue = (($config['vflip'] == 'true') || ($config['vflip'] == 1) ? 2:0);
      $cvalue += (($config['hflip'] == 'true') || ($config['hflip'] == 1) ? 1:0);
      break;
    case 'MP4Box':
      $cvalue = $config[$selKey];
      if ($cvalue == 'background') $cvalue = 2;
      break;
    default: $cvalue = $config[$selKey]; break;
```

```
  }
  if ($cvalue == 'false') $cvalue = 0;
  else if ($cvalue == 'true') $cvalue = 1;
  foreach($options as $name => $value) {
    if ($cvalue != $value) {
      $selected = '';
    } else {
      $selected = ' selected';
    }
    echo "<option value='$value'$selected>$name</option>";
  }
}

function makeInput($id, $size, $selKey='') {
  global $config, $debugString;
  if ($selKey == '') $selKey = $id;
  switch ($selKey) {
    case 'tl_interval':
      if (array_key_exists($selKey, $config)) {
        $value = $config[$selKey] / 10;
      } else {
        $value = 3;
      }
      break;
    case 'watchdog_interval':
      if (array_key_exists($selKey, $config)) {
        $value = $config[$selKey] / 10;
      } else {
        $value = 0;
      }
      break;
    default: $value = $config[$selKey]; break;
  }
  echo "<input type='text' size=$size id='$id' value='$value'>";
}

function getImgWidth() {
  global $config;
  if($config['vector_preview'])
    return 'style="width:' . $config['width'] . 'px;"';
  else
    return '';
}

function getLoadClass() {
  global $config;
  if(array_key_exists('fullscreen', $config) && $config['fullscreen'] == 1)
    return 'class="fullscreen" ';
  else
    return '';
}

if (isset($_POST['extrastyle'])) {
  if (file_exists('css/' . $_POST['extrastyle'])) {
    $fp = fopen(BASE_DIR . '/css/extrastyle.txt', "w");
    fwrite($fp, $_POST['extrastyle']);
    fclose($fp);
  }
}

$toggleButton = "Simple";
$displayStyle = 'style="display:block;"';
if(isset($_COOKIE["display_mode"])) {
```

```
    if($_COOKIE["display_mode"] == "Simple") {
      $toggleButton = "Full";
      $displayStyle = 'style="display:none;"';
    }
  }

  $streamButton = "MJPEG-Stream";
  $mjpegmode = 0;
  if(isset($_COOKIE["stream_mode"])) {
    if($_COOKIE["stream_mode"] == "MJPEG-Stream") {
      $streamButton = "Default-Stream";
      $mjpegmode = 1;
    }
  }
  $config = readConfig($config, CONFIG_FILE1);
  $config = readConfig($config, CONFIG_FILE2);
  $video_fps = $config['video_fps'];
  $divider = $config['divider'];
 ?>

<html>
  <head>
    <meta name="viewport" content="width=550, initial-scale=1">
    <title><?php echo CAM_STRING; ?></title>
    <link rel="stylesheet" href="css/style_minified.css" />
    <link rel="stylesheet" href="<?php echo getStyle(); ?>" />
    <script src="js/style_minified.js"></script>
    <script src="js/script.js"></script>
    <script src="js/pipan.js"></script>
  </head>
  <body onload="setTimeout('init(<?php echo "$mjpegmode, $video_fps, $divider" ?>);', 100);">
            <!--GPIO18-->
            <form action="" method="post">
            GPIO 18 
           <input type="submit" name="inicio" value="inicio">
           <input type="submit" name="10" value="10">
           <input type="submit" name="20" value="20">
           <input type="submit" name="30" value="30">
           <input type="submit" name="40" value="40">
           <input type="submit" name="50" value="50">
           <input type="submit" name="60" value="60">
           <input type="submit" name="70" value="70">

            <br></br>
            <!--GPIO19-->
            <form action="" method="post">
            GPIO 19 
      <input type="submit" name="centro" value="centro">
      <input type="submit" name="5" value="5">
      <input type="submit" name="15" value="15">
      <input type="submit" name="25" value="25">
      <input type="submit" name="35" value="35">
      <input type="submit" name="45" value="45">
      <input type="submit" name="-5" value="-5">
      <input type="submit" name="-15" value="-15">
      <input type="submit" name="-25" value="-25">
      <input type="submit" name="-35" value="-35">
      <input type="submit" name="-45" value="-45">

          <br></br>

          <?php
          if ($_POST[inicio]) {
```

```
      for ($i= 0; $i<10; $i++){
                  $a- exec("sudo python /var/www/html/servos/gpio/18/inicio.py");
                  echo $a;
                  }
       }
if ($_POST[10]) {
      for ($i= 0; $i<10; $i++){
        $a- exec("sudo python /var/www/html/servos/gpio/18/10.py");
                   echo $a;
       }
       }
if ($_POST[20]) {
      for ($i= 0; $i<10; $i++){
       $a- exec("sudo python /var/www/html/servos/gpio/18/20.py");
       echo $a;
                 }
 }
if ($_POST[30]) {
      for ($i= 0; $i<10; $i++){
        $a- exec("sudo python /var/www/html/servos/gpio/18/30.py");
       echo $a;
      }
       }
if ($_POST[40]) {
      for ($i= 0; $i<10; $i++){
        $a- exec("sudo python /var/www/html/servos/gpio/18/40.py");
         echo $a;
       }
       }
if ($_POST[50]) {
      for ($i= 0; $i<10; $i++){
        $a- exec("sudo python /var/www/html/servos/gpio/18/50.py");
       echo $a;
                 }
 }
if ($_POST[60]) {
      for ($i= 0; $i<10; $i++){
       $a- exec("sudo python /var/www/html/servos/gpio/18/60.py");
        echo $a;
                 }
 }
if ($_POST[70]) {
                  for ($i= 0; $i<10; $i++){
            $a- exec("sudo python /var/www/html/servos/gpio/18/70.py");
         echo $a;
       }
       }
        if ($_POST[apagar18]) {
         $a- exec("sudo python /var/www/html/servos/gpio/18/apagar.py");
         echo $a;
        }
        if ($_POST[-45]) {
      for ($i= 0; $i<10; $i++){
                   $a- exec("sudo python /var/www/html/servos/gpio/19/-45.py");
                   echo $a;
                  }
       }
 if ($_POST[-35]) {
      for ($i= 0; $i<10; $i++){
            $a- exec("sudo python /var/www/html/servos/gpio/19/-35.py");
                   echo $a;
                  }
 }
```

```
    if ($_POST[-25]) {
        for ($i= 0; $i<10; $i++){
                $a- exec("sudo python /var/www/html/servos/gpio/19/-25.py");
                echo $a;
                    }
    }
    if ($_POST[-15]) {
        for ($i= 0; $i<10; $i++){
                $a- exec("sudo python /var/www/html/servos/gpio/19/-15.py");
                echo $a;
                    }
    }
    if ($_POST[-5]) {
        for ($i= 0; $i<10; $i++){
                $a- exec("sudo python /var/www/html/servos/gpio/19/-5.py");
                echo $a;
                    }
    }
    if ($_POST[centro]) {
        for ($i= 0; $i<10; $i++){
                $a- exec("sudo python /var/www/html/servos/gpio/19/inicio.py");
                echo $a;
                    }
    }
    if ($_POST[5]) {
        for ($i= 0; $i<10; $i++){
                $a- exec("sudo python /var/www/html/servos/gpio/19/5.py");
                echo $a;
                    }
    }
    if ($_POST[15]) {
        for ($i= 0; $i<10; $i++){
                $a- exec("sudo python /var/www/html/servos/gpio/19/15.py");
                echo $a;
                    }
    }
    if ($_POST[25]) {
        for ($i= 0; $i<10; $i++){
                $a- exec("sudo python /var/www/html/servos/gpio/19/25.py");
                echo $a;
                    }
    }
    if ($_POST[35]) {
        for ($i= 0; $i<10; $i++){
                $a- exec("sudo python /var/www/html/servos/gpio/19/35.py");
                echo $a;
                    }
    }
    if ($_POST[45]) {
        for ($i= 0; $i<10; $i++){
                $a- exec("sudo python /var/www/html/servos/gpio/19/45.py");
                echo $a;
                    }
    }
          ?>

  <div class="navbar navbar-inverse navbar-fixed-top" role="navigation" <?php echo $displayStyle; ?>>
    <div class="container">
      <div class="navbar-header">
        <a class="navbar-brand" href="#"><?php echo CAM_STRING; ?></a>
      </div>
    </div>
  </div>
```

```
    <input id="toggle_display" type="button" class="btn btn-primary" value="<?php echo $toggleButton; ?>"
style="position:absolute;top:60px;right:10px;" onclick="set_display(this.value);">
    <div class="container-fluid text-center liveimage">
      <div><img id="mjpeg_dest" <?php echo getLoadClass() . getImgWidth();?> <?php if(file_exists("pipan_on"))
echo "ontouchstart=\"pipan_start()\""; ?> onclick="toggle_fullscreen(this);" src="./loading.jpg"></div>
      <div id="main-buttons" <?php echo $displayStyle; ?> >
        <input id="video_button" type="button" class="btn btn-primary">
        <input id="image_button" type="button" class="btn btn-primary">
        <input id="timelapse_button" type="button" class="btn btn-primary">
        <input id="md_button" type="button" class="btn btn-primary">
        <input id="halt_button" type="button" class="btn btn-danger">
      </div>
    </div>
    <div id="secondary-buttons" class="container-fluid text-center" <?php echo $displayStyle; ?> >
      <?php pan_controls(); ?>
      <a href="preview.php" class="btn btn-default">Download Videos and Images</a>
        
      <?php      if($config['motion_external']): ?><a href="motion.php" class="btn btn-default">Edit motion
settings</a>  <?php endif; ?>
      <a href="schedule.php" class="btn btn-default">Edit schedule settings</a>
    </div>

    <div class="container-fluid text-center">
      <div class="panel-group" id="accordion" <?php echo $displayStyle; ?> >
        <div class="panel panel-default">
          <div class="panel-heading">
            <h2 class="panel-title">
              <a data-toggle="collapse" data-parent="#accordion" href="#collapseOne">Camera Settings</a>
            </h2>
          </div>
          <div id="collapseOne" class="panel-collapse collapse">
            <div class="panel-body">
              <table class="settingsTable">
                <tr>
                  <td>Resolutions:</td>
                  <td>Load Preset: <select onchange="set_preset(this.value)">
                      <option value="1920 1080 25 25 2592 1944">Select option...</option>
                      <option value="1920 1080 25 25 2592 1944">Full HD 1080p 16:9</option>
                      <option value="1280 0720 25 25 2592 1944">HD-ready 720p 16:9</option>
                      <option value="1296 972 25 25 2592 1944">Max View 972p 4:3</option>
                      <option value="768 576 25 25 2592 1944">SD TV 576p 4:3</option>
                      <option value="1920 1080 01 30 2592 1944">Full HD Timelapse (x30) 1080p 16:9</option>
                    </select><br>
                    Custom Values:<br>
                    Video res: <?php makeInput('video_width', 4); ?>x<?php makeInput('video_height', 4); ?>px<br>
                    Video  fps:  <?php  makeInput('video_fps',  2);  ?>recording,  <?php  makeInput('MP4Box_fps',  2);
?>boxing<br>
                    Image res: <?php makeInput('image_width', 4); ?>x<?php makeInput('image_height', 4); ?>px<br>
                    <input type="button" value="OK" onclick="set_res();">
                  </td>
                </tr>
                <?php  if($config['camera_num'] > 0): ?>
                <tr>
                  <td>Camera select (Compute module only)</td>
                  <td>
                    Use  camera:  <select  onchange="send_cmd('cn  ' + this.value)"><?php  makeOptions($options_cn,
'camera_num'); ?></select>
                  </td>
                </tr>
                <?php endif; ?>
                <tr>
                  <td>Timelapse-Interval (0.1...3200):</td>
```

```
                <td><?php makeInput('tl_interval', 4); ?>s <input type="button" value="OK" onclick="send_cmd('tv '
+ 10 * document.getElementById('tl_interval').value)"></td>
              </tr>
              <tr>
                <td>Annotation (max 127 characters):</td>
                <td>
                  Text: <?php makeInput('annotation', 20); ?><input type="button" value="OK"
onclick="send_cmd('an ' + encodeURI(document.getElementById('annotation').value))"><input type="button"
value="Default" onclick="document.getElementById('annotation').value = 'RPi Cam %Y.%M.%D_%h:%m:%s';
send_cmd('an ' + encodeURI(document.getElementById('annotation').value))"><br>
                  Background: ><select onchange="send_cmd('ab ' + this.value)"><?php makeOptions($options_ab,
'anno_background'); ?></select>
                </td>
              </tr>
              <?php if (file_exists("pilight_on")) pilight_controls(); ?>
              <tr>
                <td>Buffer (1000... ms), default 0:</td>
                <td><?php makeInput('video_buffer', 4); ?><input type="button" value="OK" onclick="send_cmd('bu
' + document.getElementById('video_buffer').value)"></td>
              </tr>                <tr>
                <td>Sharpness (-100...100), default 0:</td>
                <td><?php makeInput('sharpness', 4); ?><input type="button" value="OK" onclick="send_cmd('sh ' +
document.getElementById('sharpness').value)"></td>
              </tr>
              <tr>
                <td>Contrast (-100...100), default 0:</td>
                <td><?php makeInput('contrast', 4); ?><input type="button" value="OK" onclick="send_cmd('co ' +
document.getElementById('contrast').value)">
                </td>
              </tr>
              <tr>
                <td>Brightness (0...100), default 50:</td>
                <td><?php makeInput('brightness', 4); ?><input type="button" value="OK" onclick="send_cmd('br ' +
document.getElementById('brightness').value)"></td>
              </tr>
              <tr>
                <td>Saturation (-100...100), default 0:</td>
                <td><?php makeInput('saturation', 4); ?><input type="button" value="OK" onclick="send_cmd('sa ' +
document.getElementById('saturation').value)"></td>
              </tr>
              <tr>
                <td>ISO (100...800), default 0:</td>
                <td><?php makeInput('iso', 4); ?><input type="button" value="OK" onclick="send_cmd('is ' +
document.getElementById('iso').value)"></td>
              </tr>
              <tr>
                <td>Metering Mode, default 'average':</td>
                <td><select onchange="send_cmd('mm ' + this.value)"><?php makeOptions($options_mm,
'metering_mode'); ?></select></td>
              </tr>
              <tr>
                <td>Video Stabilisation, default: 'off'</td>
                <td><select onchange="send_cmd('vs ' + this.value)"><?php makeOptions($options_vs,
'video_stabilisation'); ?></select></td>
              </tr>
              <tr>
                <td>Exposure Compensation (-10...10), default 0:</td>
                <td><?php makeInput('exposure_compensation', 4); ?><input type="button" value="OK"
onclick="send_cmd('ec ' + document.getElementById('exposure_compensation').value)"></td>
              </tr>
              <tr>
                <td>Exposure Mode, default 'auto':</td>
```

```
                <td><select onchange="send_cmd('em ' + this.value)"><?php makeOptions($options_em, 'exposure_mode'); ?></select></td>
              </tr>
              <tr>
                <td>White Balance, default 'auto':</td>
                <td><select onchange="send_cmd('wb ' + this.value)"><?php makeOptions($options_wb, 'white_balance'); ?></select></td>
              </tr>
              <tr>
                <td>White Balance Gains (x100):</td>
                <td> gain_r <?php makeInput('ag_r', 4, 'autowbgain_r'); ?> gain_b <?php makeInput('ag_b', 4, 'autowbgain_b'); ?>
                  <input type="button" value="OK" onclick="set_ag();">
                </td>
              </tr>
              <tr>
                <td>Image Effect, default 'none':</td>
                <td><select onchange="send_cmd('ie ' + this.value)"><?php makeOptions($options_ie, 'image_effect'); ?></select></td>
              </tr>
              <tr>
                <td>Colour Effect, default 'disabled':</td>
                <td><select id="ce_en"><?php makeOptions($options_ce_en, 'colour_effect_en'); ?></select>
                  u:v = <?php makeInput('ce_u', 4, 'colour_effect_u'); ?>:<?php makeInput('ce_v', 4, 'colour_effect_v'); ?>
                  <input type="button" value="OK" onclick="set_ce();">
                </td>
              </tr>
              <tr>
                <td>Image Statistics, default 'Off':</td>
                <td><select onchange="send_cmd('st ' + this.value)"><?php makeOptions($options_st, 'stat_pass'); ?></select></td>
              </tr>
              <tr>
                <td>Rotation, default 0:</td>
                <td><select onchange="send_cmd('ro ' + this.value)"><?php makeOptions($options_ro, 'rotation'); ?></select></td>
              </tr>
              <tr>
                <td>AngleX (-45...45), default 0:</td>
                <td><?php makeInput('anglex', 4); ?><input type="button" value="OK" onclick="send_cmd('ax ' + document.getElementById('anglex').value)">
                </td>
              </tr>
              <tr>
                <td>AngleY (0...75), default 0:</td>
                <td><?php makeInput('angley', 4); ?><input type="button" value="OK" onclick="send_cmd('ay ' + document.getElementById('angley').value)">
                </td>
              </tr>
              <tr>
                <td>Flip, default 'none':</td>
                <td><select onchange="send_cmd('fl ' + this.value)"><?php makeOptions($options_fl, 'flip'); ?></select></td>
              </tr>
              <tr>
                <td>Sensor Region, default 0/0/65536/65536:</td>
                <td>
                  x<?php makeInput('roi_x', 5, 'sensor_region_x'); ?> y<?php makeInput('roi_y', 5, 'sensor_region_y'); ?> w<?php makeInput('roi_w', 5, 'sensor_region_w'); ?> h<?php makeInput('roi_h', 4, 'sensor_region_h'); ?> <input type="button" value="OK" onclick="set_roi();">
                </td>
              </tr>
```

```
                <tr>
                  <td>Shutter speed (0...330000), default 0:</td>
                  <td><?php makeInput('shutter_speed', 4); ?><input type="button" value="OK"
onclick="send_cmd('ss ' + document.getElementById('shutter_speed').value)">
                  </td>
                </tr>
                <tr>
                  <td>Image quality (0...100), default 10:</td>
                  <td>
                    <?php makeInput('image_quality', 4); ?><input type="button" value="OK" onclick="send_cmd('qu '
+ document.getElementById('image_quality').value)">
                  </td>
                </tr>
                <tr>
                  <td>Preview quality (1...100) Default 10:<br>Width (128...1024) Default 512:<br>Divider (1-16)
Default 1:</td>
                  <td>
                    Qu: <?php makeInput('quality', 4); ?>
                    Wi: <?php makeInput('width', 4); ?>
                    Di: <?php makeInput('divider', 4); ?>
                    <input type="button" value="OK" onclick="set_preview();">
                  </td>
                </tr>
                <tr>
                  <td>Raw Layer, default: 'off'</td>
                  <td><select onchange="send_cmd('rl ' + this.value)"><?php makeOptions($options_rl, 'raw_layer');
?></select></td>
                </tr>
                <tr>
                  <td>Video bitrate (0...25000000), default 17000000:</td>
                  <td>
                    <?php makeInput('video_bitrate', 10); ?><input type="button" value="OK" onclick="send_cmd('bi '
+ document.getElementById('video_bitrate').value)">
                  </td>
                </tr>
                <tr>
                  <td>MP4 Boxing mode :</td>
                  <td><select onchange="send_cmd('bo ' + this.value)"><?php makeOptions($options_bo, 'MP4Box');
?></select></td>
                </tr>
                <tr>
                  <td>Annotation size(0-99):</td>
                  <td>
                    <?php makeInput('anno_text_size', 3); ?><input type="button" value="OK" onclick="send_cmd('as '
+ document.getElementById('anno_text_size').value)">
                  </td>
                </tr>
                <tr>
                  <td>Custom text color:</td>
                  <td><select id="at_en"><?php makeOptions($options_at_en, 'anno3_custom_text_colour');
?></select>
                    y:u:v = <?php makeInput('at_y', 3, 'anno3_custom_text_Y'); ?>:<?php makeInput('at_u', 4,
'anno3_custom_text_U'); ?>:<?php makeInput('at_v', 4, 'anno3_custom_text_V'); ?>
                    <input type="button" value="OK" onclick="set_at();">
                  </td>
                </tr>
                <tr>
                  <td>Custom background color:</td>
                  <td><select id="ac_en"><?php makeOptions($options_ac_en, 'anno3_custom_background_colour');
?></select>
                    y:u:v = <?php makeInput('ac_y', 3, 'anno3_custom_background_Y'); ?>:<?php makeInput('ac_u', 4,
'anno3_custom_background_U'); ?>:<?php makeInput('ac_v', 4, 'anno3_custom_background_V'); ?>
                    <input type="button" value="OK" onclick="set_ac();">
```

```
                </td>
                </tr>
              <tr>
                <td>Watchdog, default interval 3s, errors 3</td>
                <td>Interval <?php makeInput('watchdog_interval', 3); ?>s    Errors <?php
makeInput('watchdog_errors', 3); ?>
                <input type="button" value="OK" onclick="send_cmd('wd ' + 10 *
document.getElementById('watchdog_interval').value + ' ' + document.getElementById('watchdog_errors').value)">
                </td>
              </tr>
              <tr>
                <td>Motion detect mode :</td>
                <td><select onchange="send_cmd('mx ' + this.value);setTimeout(function(){location.reload(true);},
1000);"><?php makeOptions($options_mx, 'motion_external'); ?></select></td>
              </tr>
            </table>
          </div>
        </div>
      </div>
      <div class="panel panel-default" <?php if($config['motion_external']) echo "style ='display:none;'"; ?>>
        <div class="panel-heading">
          <h2 class="panel-title">
            <a data-toggle="collapse" data-parent="#accordion" href="#collapseTwo">Motion Settings</a>
          </h2>
        </div>
        <div id="collapseTwo" class="panel-collapse collapse">
          <div class="panel-body">
            <table class="settingsTable">
              <tr>
                <td>Motion Vector Preview:</td>
                <td>
                 <select onchange="send_cmd('vp ' + this.value);setTimeout(function(){location.reload(true);},
1000);" id="preview_select"><?php makeOptions($options_vp, 'vector_preview'); ?></select>
                </td>
              </tr>
              <tr>
                <td>Noise level (1-255):</td>
                <td>
                  <?php makeInput('motion_noise', 5); ?><input type="button" value="OK" onclick="send_cmd('mn '
+ document.getElementById('motion_noise').value)">
                </td>
              </tr>
              <tr>
                <td>Threshold (1-32000):</td>
                <td>
                  <?php makeInput('motion_threshold', 5); ?><input type="button" value="OK"
onclick="send_cmd('mt ' + document.getElementById('motion_threshold').value)">
                </td>
              </tr>
              <tr>
                <td>Mask Image:</td>
                <td>
                  <?php makeInput('motion_image', 30); ?><input type="button" value="OK" onclick="send_cmd('mi
' + document.getElementById('motion_image').value)">
                </td>
              </tr>
              <tr>
                <td>Change Frames to start:</td>
                <td>
                  <?php makeInput('motion_startframes', 5); ?><input type="button" value="OK"
onclick="send_cmd('mb ' + document.getElementById('motion_startframes').value)">
                </td>
              </tr>
```

```
                <tr>
                  <td>Still Frames to stop:</td>
                  <td>
                    <?php makeInput('motion_stopframes', 5); ?><input type="button" value="OK" onclick="send_cmd('me ' + document.getElementById('motion_stopframes').value)">
                  </td>
                </tr>
                <tr>
                  <td>Save vectors to .dat :<br>Uses more space</td>
                  <td><select onchange="send_cmd('mf ' + this.value);"><?php makeOptions($options_mf, 'motion_file'); ?></select></td>
                </tr>
              </table>
            </div>
          </div>
        </div>
        <div class="panel panel-default">
          <div class="panel-heading">
            <h2 class="panel-title">
              <a data-toggle="collapse" data-parent="#accordion" href="#collapseThree">System</a>
            </h2>
          </div>
          <div id="collapseThree" class="panel-collapse collapse">
            <div class="panel-body">
              <input id="toggle_stream" type="button" class="btn btn-primary" value="<?php echo $streamButton; ?>" onclick="set_stream_mode(this.value);">
              <input id="shutdown_button" type="button" value="shutdown system" onclick="sys_shutdown();" class="btn btn-danger">
              <input id="reboot_button" type="button" value="reboot system" onclick="sys_reboot();" class="btn btn-danger">
              <input id="reset_button" type="button" value="reset settings" onclick="send_cmd('rs 1');setTimeout(function(){location.reload(true);}, 1000);" class="btn btn-danger">
              <form action='<?php echo ROOT_PHP; ?>' method='POST'>
                <br>Style
                <select name='extrastyle' id='extrastyle'>
                  <?php getExtraStyles(); ?>
                </select>
                 <button type="submit" name="OK" value="OK" >OK</button>
              </form>
            </div>
          </div>
        </div>
      </div>
    </div>
    <?php if ($debugString != "") echo "$debugString<br>"; ?>
  </body>
</html>
```

Anexo 2. Octorpint

Comandos octoprint:

Gnnn	Comando GCode estándar, como moverse hasta un punto
Xnnn	Una coordenada X, normalmente para moverse a ella. Puede ser un número entero o racional.
Ynnn	Una coordenada Y, normalmente para moverse a ella. Puede ser un número entero o racional.
Znnn	Una coordenada Z, normalmente para moverse a ella. Puede ser un número entero o racional
Fnnn	Feedrate en mm por minuto. (Velocidad de movimiento del cabezal de impresión)
Rnnn	Parámetro - usado para temperaturas

G0 Xnnn Ynnn Znnn Ennn Fnnn Snnn

G1 Xnnn Ynnn Znnn Ennn Fnnn Snnn

Xnnn The position to move to on the X axis

Ynnn The position to move to on the Y axis

Znnn The position to move to on the Z axis

Ennn The amount to extrude between the starting point and ending point

Fnnn The feedrate per minute of the move between the starting point and ending point (if supplied)

Snnn Flag to check if an endstop was hit *(**S1** to check, **S0** to ignore, **S2** see note, default is **S0**)*

1.1.1.1 G2 & G3: Controlled Arc Move

G2 Xnnn Ynnn Innn Jnnn Ennn *(Clockwise Arc)*

G3 Xnnn Ynnn Innn Jnnn Ennn *(Counter-Clockwise Arc)*

Xnnn The position to move to on the X axis

Ynnn The position to move to on the Y axis

Innn The point in X space from the current X position to maintain a constant distance from

Jnnn The point in Y space from the current Y position to maintain a constant distance from

Ennn The amount to extrude between the starting point and ending point

G2 X90.6 Y13.8 I5 J10 E22.4 *(Move in a Clockwise arc from the current point to point (X=90.6,Y=13.8), with a center point at (X=current_X+5, Y=current_Y+10), extruding 22.4mm of material between starting and stopping)*

G3 X90.6 Y13.8 I5 J10 E22.4 *(Move in a Counter-Clockwise arc from the current point to point (X=90.6,Y=13.8), with a center point at (X=current_X+5, Y=current_Y+10), extruding 22.4mm of material between starting and stopping)*

1.1.1.2 G90: Set to Absolute Positioning

All coordinates from now on are absolute relative to the origin of the machine. (This is the RepRap default.)

1.1.1.3 G91: Set to Relative Positioning

All coordinates from now on are relative to the last position.

1.1.1.4 G92: Set Position

Example: G92 X10 E90

Allows programming of absolute zero point, by reseting the current position to the values specified. This would set the machine's X coordinate to 10, and the extrude coordinate to 90. No physical motion will occur.

G28: Move to Origin (Home)

	FiveD	Teacup	Sprinter	Marlin	Repetier	Smoothie	RepRapFirmware
Support	yes	yes	yes	yes	yes	yes	yes

Usage

G28

Variables

This Gcode can be used without any additional variables supplied

X Flag to go back to the X axis origin

Y Flag to go back to the Y axis origin

Z Flag to go back to the Z axis origin

Examples

G28 *(Go to origin on all axes)*

G28 X Z *(Go to origin only on the X and Z axis)*

Para más información acerca de los comandos se puede acceder a la siguiente página:

http://reprap.org/wiki/G-code/es

Se va a proporcionar un enlace en el cual se explica la función de cada botón en la página octoprint:

http://docs.octoprint.org/en/master/api/printer.html